Representation and Reasoning for Complex Spatio-Temporal Problems: From Humans to Software Agents

A Dissertation
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Christopher Baylor Wetzel

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Maria Gini, Wilma Koutstaal

March 2014

Acknowledgements

Lots of people say in their acknowledgements that they wouldn't be here had it not been for the support of others, but I suspect that many of those people would have succeeded no matter what. That is not the case here. I don't know if there is a normal path for a Ph.D. but I'm fairly certain that if there is, mine was far from it. I was by no means the easiest graduate student to work with and my choice of topic, interdisciplinary and novel but, more than anything else, just plain different, was not the easiest topic to advise. Despite my best efforts, my research is done, this document is written and I am a doctor. Let me tell you why.

When I began, I was simply taking a few classes for fun while working a full time job and raising a family. For a person like me, going back to any school, much less a Ph.D. program in computer science, seemed unrealistic. Actually, impossible. Maria was the one who coaxed me into becoming a student. Paula made it possible for me go back. Vipin made it possible for me to stay in. It wasn't easy and there were many times I wanted to quit. Wilma made me want to keep trying, Maria simply wouldn't let me quit. Eventually I ended up with my committee of four: Maria, my computer science adviser; Wilma, psychology adviser; Vipin, who said "Don't worry, I'll take care of it" so many times that I have no idea just how much I owe him and of course John, my committee chair who is a knowledgable, wise and caring man despite his unnatural love of puns.

Outside academia, Paula was a sounding board and cheerleader. Erin read every single page, multiple times, to make sure the text flowed as well as my skills and constant changes would allow. The research discussed here made heavy use of a tool custom built by Kyle. Every time I said something such as "I wish we could run all these AI 10,000 times in an overnight batch job and collect all the statistics", I'd wake up the next morning with an email from Kyle telling me he'd done it.

My home in psychology has treated me well. The first class I took at Minnesota was Bruce's 2003 Animal Learning course. It's been more than a decade and he still sends me papers out of the blue because he thinks it will help my research. My RA Laura did a fantastic job running experiments, collecting data and coding it. Wilma claims that the numerous weekends and late nights she spent helping analyze data and design experiements were no big deal, but she's wrong, they were.

I taught a few years at Brown College before becoming a full time student. As a student, Maria would often let me give lectures in her AI class. Rather surprisingly, when she went on sabbatical, Phil and Maria gave the class to me. The next semester, Phil and Vipin offered me a full time job. In my mind, it was always Maria's class and I was just holding onto it for her. Each semester I thought she'd ask for her it back but she never did. That's a lot of trust. I've been teaching for six years now (two at Brown, four at Minnesota) and have taught more than 1,000 students. Possibly much more – I taught 800 this year alone. A lot of those students have taken as much interest in my life as I have in theirs. Teaching, raising a family, dealing with major life events and working on a dissertation was stressful. I somehow accumulated a cheer squad, people who would literally ask me every week how things were going and could they in any way help. So here's thanks to Adeeb, Andrew, Brian, Buyu, Chad, Cindy, Cody, Dominick, Eitan, Emily, Eric, Ethan, Fang Yi, Grant, Hien, Iustis, Ivan, James, Jessie, Jessica, Jonathan, Justin, Karel, Kevin, Kristen, Laura, Logain, Magnum, Matt L., Matt O., Mia, Michael, Nick, Noah, Ollie, Patrick, the other Patrick, Pratheeksha, Randall, Reagan, Ricky, Roman, Ryan, Shraddha, Taylor and Zach. And probably several others that will cause me to be horrified when I realize I left this off this list. Outside of school, my biggest cheerleaders have probably been Brock and Kevin, to the extent that "just shut up and do it" is cheering. Thanks guys. I also can't forget Andre, who was convinced I'd make a great researcher, to the point that he offered me a job at his company, and Ian, who is far more empathetic and friendly than any professor has a right to be.

Dedication

This work is as much Paula's as mine, at least the good parts. It would be impossible to list all the ways she made this possible. Beyond financial and emotional support, she's a world class programmer and (graduated) graduate student who went to every conference with me, helped me with every project and provided advice every time I got stuck. She even wrangled Niko and Kale into helping out. She is, unfortunately, the uncredited star, the person behind the scenes who makes everything work. She lets me take the credit but she knows as well as I do that there is no way I could or would have done this without her. In a perfect world, we would be sharing our new degree and she would henceforth be known as Dr. Paula.

As for the girls… You were just starting school when I did and now here we are, all in college together. That's kind of embarrassing. I'm not sure you remember what it was like to have a daddy that wasn't a student. I finished just in time for you to leave home. Well, almost in time. It was a Cat's in the Cradle kind of life. Sorry about that. By the way, while we're doing confessions, I spent your inheritance on college, so in exchange, please enjoy this fine dissertation – it's got pictures and tables and everything!

Abstract

Success in the real world depends on the ability to reason about space and time. Consider the simple, everyday task of deciding whether to cross a road. If a car is coming, your decision will be based on how wide the road is, how fast you walk, how far away the car is and how fast it is moving. You might also consider structural features of the road; if the car has to turn a corner or go over a speed bump, the car will move slower for a short period of time. Determining whether it is safe to cross requires reasoning about the interaction of these variables.

The field of artificial intelligence has developed representations for describing space (e.g., RCC) and time (e.g., interval calculus) but not for describing the interaction between the two. Unsurprisingly, billions of years of evolution have resulted in humans being quite good at it. How they do so is not completely understood. Detailed studies have focused on overly simple problems while studies of complex problems have lacked sufficient detail to build computer models.

This thesis describes our investigation into solving problems with significant spatio-temporal components. We focused on the domain of tower defense puzzles, a class of complex spatio-temporal problems that requires the problem solver to use spatial actions (placing guard towers on a map) to maximize a temporal variable (tower active time). We had two objectives. First, using methods from experimental psychology and computational behavioral modeling, we wished to understand how, precisely, humans solved these problems. Humans, unlike computers, are known to be good at solving this type of problem. Our second goal was to construct a computer agent capable of solving this task as well as or better than the best humans.

To investigate the relationship of space, time and problem solving, we performed two experiments. The goal of Experiment 1 was to determine how humans solved tower defense puzzles. Experienced tower defense solvers (n=38) were asked to solve a series of novel tower defense puzzles. Interviews and automated data capture tools provided data that were used to answer a set of questions on how humans solved these puzzles. The results showed a tight integration between problem, representation and reasoning. Subjects needed to manipulate space to maximize a temporal value. Rather than represent the space, they represented the goal-relevant opportunities for actions present in the space, known as affordances. Problems were decomposed into sets of goals, each goal was addressed by one or more simple, focused, goal-specific strategies and each strategy was activated by an affordance. An interesting finding was that many subjects treated temporal problems as if they were spatial ones, which we refer to as spatial proxying.

The goal of Experiment 2 was to determine how well the discovered strategies worked. Novice (n=10) and experienced (n=10) tower defense solvers were asked to solve a series of novel tower defense puzzles. Results showed that 70% of novice and 40% of experienced subjects used spatial proxying strategies and that these strategies worked surprisingly well. 10% of novice and 60% of experienced subjects used strategies that directly manipulated time. These strategies performed better but frequently created solutions that were counter-intuitive.

Our second objective was to investigate computational representations and algorithms capable of creating an agent that performs at or above human levels for this task. Human studies showed that the majority of the "intelligence" of their problem solving process lay in the recognition and representation of spatial affordances. This led to the creation of the Spatial Affordance Query System (SAQS). In this system, spatio-temporal reasoning agents are created declaratively, with the author specifying the strategies the agent knows. The agent and problem map are passed to a solver, which compares the agent's strategy set to the affordances reported by SAQS, which instantiates the applicable strategies. The majority of agents were 5-12 lines of code and the best agent performed at the same level as the best human subjects.

Table of Contents

List of Tables

List of Figures

List of Code Listings

List of Strategies

Chapter 1

Introduction

1.1 Motivation

Success in the real world requires understanding space, time and, most importantly, the interaction of the two. A simple example is crossing a road. Spatial reasoning tells you how to cross the road and temporal reasoning tells you how long it will take but you need integrated spatio-temporal reasoning to cross a road without being hit by a car. Reasoning about how objects move through space through time – where a car will be when you make it to the middle of the road and how a corner will affect the speed and location of the car – is critical to tasks as small as catching a ball and intercepting a criminal and as large as fighting a war and laying out a city.

Computer science has generated many methods for representing and reasoning about space and time but relatively few regarding the integration of the two (Kreutzmann and Wolter 2011). Most of the work has focused on generic, re-usable methods that work in any domain for any task. These methods work well for describing space and time and reasoning about relationships in those terms (e.g., if A is before B and B is before C, is A before C?) but as prevalent as space and time are in real world tasks, the goals of real world tasks are rarely spatio-temporal. Pedestrians want to know if they can cross the street safely. Urban planners want to know how the placement of roads and shopping centers will affect the growth of a neighborhood. Armies want to know if they are likely to be trapped in a bottleneck. Representing the interaction of objects moving through space through time to extract non-spatio-temporal information in pursuit of non-spatio-temporal goals is important to survival in the real world and, as a consequence, is a fundamental capability of all creatures that exist in it. For computers to reason about or interact with the real world, it is important for computers to develop this capability.

Reasoning about spatio-temporal interactions is a surprisingly difficult problem. We know humans can do it but do not understand precisely how it works. Likewise, understanding how spatio-temporal reasoning contributes to human problem solving requires understanding how humans solve problems. A considerable amount of effort has gone into studying human problem solving, both in psychology and cognitive science (Anderson et al. 2004), but the research on solving real-world problems (e.g., Klein 1998) has been at too high a level of abstraction to model computationally while tasks that have been modeled at a sufficient level of detail (e.g., (Smith, John P., Disessa and Roschelle 1993; Gigerenzer and Todd 1999a) are too simple to explain how real world problems are solved.

How does one build a computer program capable of solving complex problems with significant spatio-temporal components? The approach we describe in this paper looks to human reasoning as inspiration. To do this we need to investigate several inter-related topics:

- How do humans solve complex problems?
- How do humans represent space and time?
- How do humans use this information to solve complex spatio-temporal problems?
- How well do human methods work? Why?

- What lessons can be applied to computer agents?
- How well would these computer agents perform?

The work described here uses human studies to develop computer spatio-temporal problem solving agents. Several dozen subjects were observed while solving a set of tower defense puzzles. Tower defense puzzles, explained in detail later in this paper, ask the problem solver to prevent enemies from making it through a specified area by placing a set of defensive towers at strategic locations. The towers attack enemies that come into range and the effectiveness of the towers depends on where the towers are placed. Tower defense puzzles are a class of complex spatio-temporal puzzles that humans do well on but for which currently there is no efficient computational method for solving. We analyzed data from human subjects solving sets of such puzzles to extract key concepts, representations and processes and then used those extractions to build a set of computer agents.

1.2 Contributions

The work described here is intended to contribute to psychology, cognitive science and computer science. We better elaborate on mechanisms discussed in psychology and leverage this to create new capabilities for computer agents.

1.2.1 Insights into Human Problem Solving for Complex Problems

Work in bounded rationality suggests that humans solve problems using simple, problem-focused representations and strategies. Research has focused on simple, context-independent tasks such as comparing fractions (Smith, John P., Disessa and Roschelle 1993) or judging the size of a city (Gigerenzer and Todd 1999a). Subjects analyzed each problem to determine the best representation and strategy to use and then applied the selected strategy. In this thesis we investigated problem solving in a more complex domain. What we found validates and extends previous findings. Subjects used simple, problem-focused representations and strategies selected by analyzing the features of each problem but rather than use a single strategy they used sets of strategies, re-representing the problem as a set of goals and selecting strategies for each goal. Most strategies operated independently but some were combined, with the combination determined by meta-level strategies. Some strategies modified other strategies (often in the form of exclusions, i.e., don't consider this option). Prioritization and tie-breaking strategies were used when a strategy selected more than one option and satisficing strategies helped determine how and for how long the subject applied the strategies.

Process	**Example**
1. Notice an unexpected event relevant to performance	After a corner, objects on the outside path are trailing those on the inside path. The group takes longer to pass the next towers, giving the towers more time to attack.
2. Hypothesize why the event happened	The outside path is longer than the inside one. When one group falls behind, the combined length of the groups is longer. Longer groups take longer to move past a tower.
3. Create features & affordances to support the hypothesis	Corner, inner/outer path, path length, group synchronization, group length, time in attack range
4. Create strategies to modify the frequency of these events	Place towers after corners. More abstract: Place towers where groups are most out of synch.

Table 1.1. Learning From Perception. Event perception leads to the creation of new features that, in turn, leads to the creation of new strategies.

Another contribution of this research is showing that humans solved this class of problems (and, we suspect, most other problems of this complexity) at the level of the system, with solutions emerging from a strong interdependence between learning, perception, sub-goal formation, strategy and representation (a simplified example is given in Table 1.1). Each aspect of the problem-solving process (e.g., perception, strategy, and representation) must be studied in the context of the others. For example, we found that subjects represented problems in terms of spatial affordances rather than concrete features, that the representation of the problems greatly influenced which strategies were selected and that the set of known strategies largely determined how problems were represented.

The distribution of the problem solving process across the multiple, tightly-integrated components of the system led to another discovery, that much of the problem solving process takes place outside of reasoning. Specifically, how a problem is represented appears to be at least as important as how it is reasoned about. As will be shown, when represented in the right way, developing strategies becomes a trivial task.

It is interesting to note that, although humans are flexible problem solvers able to handle a wide array of novel problems, many of the observed strategies were specific to this domain. This lack of generality is important. Combined with research showing similar results in other (albeit simpler) domains, it suggests that human flexibility in problem solving is at least partially not due to abstraction, as is often argued. Although it needs further investigation, this research suggests that the key to general-purpose problem solving might be the ability to rapidly develop sets of task-specific representations and strategies.

During this research we also gained some insight into how individuals vary in their problem solving approaches. In our studies, every subject had a unique problem solving style. These styles were comprised of a set of goals, one or more strategies per goal and a method for combining the (potentially mutually exclusive) strategies into a single approach. While there were more strategies (79) than subjects (58), some strategies were used by almost all subjects. The diversity of strategies is not the primary reason for the diversity of problem solving styles. The variance between problem solving styles was due to how the strategies were combined. To the best of our knowledge, this is the first research to show this level of variety in solving complex problems or explain individual differences as a function of strategy combination rather than strategy availability or use.

1.2.2 Spatio-Temporal Representation for Problem Solving

A large portion of this document is dedicated to cataloging spatio-temporal representations and strategies. We identified several dozen strategies and representational elements subjects used for this domain. These catalogs give a fascinating insight into how the problem solvers' minds worked and the multitude of different ways of viewing a spatio-temporal problem.

Subjects with no previous domain experience predominantly used spatial representations and strategies. In contrast, many of the subjects with domain experience used more complex spatio-temporal representations that supported better performing strategies. Section 6.4.2 Path Gap talks about one such concept, the path synchronization gap. Section 6.2 Temporal Extensions to Spatial Representations talks about an important class of spatio-temporal representations called temporal group attack windows.

1.2.3 Unifying Spatial and Temporal Measures

The domain discussed in this work, tower defense puzzles, has a non-spatio-temporal goal (maximize score) that correlates strongly with a temporal measure (maximize the total amount of time towers are active). One interesting finding was that subjects re-represented these problems as spatial problems (i.e., maximize space) and solved those. A second interesting finding was that this approach actually works quite well.

For complex spatio-temporal problems such as the ones used here, there is a relationship between space and time. Many subjects treated this as a linear relationship, which works well but is not entirely accurate. Some subjects used a more complex space-time relationship that led to actions which were unintuitive and sometimes extremely counter-intuitive. These subjects normally did better than those who used the simple linear model.

Subjects using a more effective space-time relationship could often explain what they were doing but not articulate why it worked. In this research, we analyzed subject representations, strategies and solutions to find an accurate space-time relationship. This led to the development of a new, unified, spatio-temporal measure, *al*, which is capable of simultaneously representing space and time, allowing space to be measured in time and vice-versa. This measure successfully predicts the performance of unintuitive and counter-intuitive strategies.

1.2.4 Spatial Affordance Query System

From the point of view of a strategy, affordances act as rule antecedents (e.g., if feature X exists, exploit it by doing action Y). The problem solving process works by matching the affordances needed by each strategy to the affordances present in the problem and then combining and instantiating the resulting set of strategies. In our computer model, the majority of our work and the majority of the system's complexity and intelligence existed in the process of identifying affordances. For humans, this is part of the perception process, and a good deal of human sensory-motor and neural learning is designed for this purpose.

Computers do not have an innate perceptual system. There is no intrinsic, automatic process that recognizes higher-level spatial and temporal configurations, meta-data or affordances. One of the primary outcomes of this work is the Spatial Affordance Query System (SAQS), which takes raw map geometry and allows an agent to make context-dependent affordance queries (e.g., to ask: where do the two lines of objects significantly diverge as seen from the perspective of a tower with a range of X).

This system shows one way that spatio-temporal properties can be represented as well as the complexity and diversity of spatial and spatio-temporal representations. The human approach to problem solving based on affordances and strategies also makes implementation of spatio-temporal reasoning much easier. Because so much of the intelligence is embedded in the representation, by off-loading recognition of spatio-temporal affordances to the SAQS system, most of our agents are less than a dozen lines of code.

1.2.5 Spatio-Temporal Reasoning Agents

We developed a number of agents in the course of this work. These agents tested the usability and completeness of the Spatial Affordance Query System and allowed us to explore and evaluate the effectiveness of different strategies.

Of particular importance are two agents, one of which used a spatial strategy, the other of which used a temporal strategy. These agents served as a test bed for determining the quality of different strategies and allowed us to test the effectiveness of these strategies against each other and against humans using equivalent strategies. This allowed us to draw a number of conclusions regarding the determinants of strategy effectiveness (e.g., how do different spatial properties of a map affect the performance of the strategy) and the conditions under which one strategy is superior to another.

The task that we selected, tower defense, is a complex spatio-temporal reasoning task with a non-spatio-temporal goal. In tower defense puzzles, the problem solver must place a set of defense towers on a map to prevent enemies from moving all the way through the map. This task requires the problem solver to identify strategic locations on the map. The layouts of the maps varies significantly, making each map a novel problem. Affordance- and strategy-based approaches allow humans to solve novel, complex problems in a minimal amount of time. A final contribution of this work is a spatio-temporal reasoning agent that performs as well at this task as the top human problem solvers.

1.3 Organization

Our investigation into spatio-temporal strategies pursued two goals. First, we wanted to understand human problem solving better, specifically for complex spatio-temporal problems. Chapter 4 discusses how we investigated this question. Chapter 5 gives an overview of what we found. Chapter 6 discusses in more detail the relationship we found between space and time. Chapter 7 provides a detailed discussion of the representations we observed people using to reason about these problems. Chapter 8 lists the strategies we observed people using and explains the reasoning behind them.

Our second goal was to extract lessons about how to solve complex spatio-temporal problems and use them to build high quality computer agents. In Chapter 9 we discuss our computer model for representations, strategies and spatio-temporal reasoning agents. Chapter 10 discusses how we measured the performance of two of the spatio-temporal reasoning agents we created. The results are discussed in Chapter 11.

The chapters focus on the big picture, discussing important concepts and the lessons to draw from them without presenting more detail than is necessary to understand the big picture. The appendices cover the details left out of the main text. Appendix A is a glossary of domain-specific terms used throughout this paper. Appendix B gives human performance data on all 16 puzzles used in the research presented here. Appendix C goes into additional detail about the human experiments. Appendix D discusses in more detail the results of the performance analysis reported in chapter 11. Appendix E provides additional detail of the computer spatio-temporal reasoning model introduced in chapter 9. Many of the puzzles described in this work were quite difficult. Appendix F provides sample solutions for all 16 puzzles used in this research.

Chapter 2

Related Work

The work described in this thesis focuses on solving complex spatio-temporal problems with a special focus on those problems where humans currently demonstrate a considerable advantage over computers. In this chapter we present what is currently known about those areas relevant to this research.

2.1 Models of Space and Time

In this section we discuss models of representation and reasoning for spatial domains, temporal domains and domains where there is an important relationship between the two.

A considerable body of research exists on computer models of representation and reasoning in spatial domains, with a particular emphasis on representation. Significantly less work has been done on temporal and spatio-temporal domains, although much of the spatial work applies to temporal domains. Psychological research has focused on properties of spatial, temporal and, to a lesser extent, spatio-temporal abilities but, as is common in psychological research, has done less work on precisely defining representations and reasoning processes.

2.1.1 Artificial Intelligence

Much of the spatial and temporal reasoning work in artificial intelligence has focused on describing qualitative relationships between items and using constraint satisfaction to determine whether those relationships legally describe a given set of constraints (Lee and Wolter 2011). Consider the example of a recipe for cupcakes that specifies that icing is spread over the cupcakes after the batter is mixed. The relationship between spreading the icing and mixing the batter is a temporal relationship. In this example, the relationship is qualitative – we know that icing comes after mixing but do not specify the precise amount of time between the two. This qualitative temporal relationship is used in temporal reasoning to answer questions such as "can we place sprinkles after spreading the icing but before mixing the batter?" (no) and "can we bake the batter after mixing it but before spreading the icing?" (yes).

Spatial reasoning has been used in geographical database systems (Egenhofer and Franzosa 1991; Grigni, Papadias, and Papadimitriou 1995), robot navigation (Frommberger 2008), molecular biology (Guidi and Roderick 1993) and identifying avenues of approach in military operations (Forbus, Mahoney, and Dill 2002). Many other areas have been identified as potential users of spatial reasoning including surgical assistants, oil pipeline planning and 3D circuit design (Cabalar and Santos 2006).

Using puzzles and games to study and develop AI techniques has a long history in AI. Puzzles and games (McCarthy 1990), especially video games (McCarthy 1998), are often considered ideal environments to test problem solving and knowledge representation in general (Cabalar and Santos 2011) and qualitative reasoning in particular (Williams and Kleer 1991) as they allow problems to be simplified to essential elements while maintaining a level of complexity necessary for research. It has been argued that the study of human-level intelligence benefits strongly from domains with a strong "physical world" component (Bredeweg and Struss 2004) such as spatial reasoning puzzles (Williams and Kleer 1991).

Several spatial reasoning puzzles have been studied. The Fisherman's Folly (Cabalar and Santos 2011) requires the problem solver to remove a ring from the bottom of a post, where a rope with objects on either end has been placed through the center of the post, by manipulating the rope. The CLOCK project (Forbus, Nielsen, and Faltings 1991) modeled on the gears of a clock, uses the spatial properties of shape and connectivity to determine how rotating one gear modifies the rest of the system.

While there are no benchmark problems in spatio-temporal reasoning, Kreutzmann and Wolter (Kreutzmann and Wolter 2011) proposed several puzzle-based domains including a bouncing ball puzzle, where the goal is to throw a ball such that it bounces into a given region; the video game Deflector, which involves arranging a set of rotatable mirrors to allow a laser beam to pop a balloon and the video games Crazy Machines and Incredible Machines, in which objects are aligned to create complex Rube Goldberg-like devices. Wintermute and Laird (Wintermute and Laird 2008) proposed several problems including predicting the position of a set of blocks initially placed in an unstable configuration after gravity is applied and predicting whether it is safe to cross a road given a car moving towards the intersection at a given speed.

The cognitive science community has investigated computer models of human behavior and cognition, including models of spatial reasoning. Well-known computational cognitive models include ACT-R (Anderson 2007; Anderson et al. 2004) and Soar (Laird, Newell, and Rosenbloom 1987). ACT-R has no innate spatial reasoning support (Lathrop and Laird 2007) but has acted as the base for systems investigating human spatial inferences (Ragni and Brüssow 2011) and applied spatial reasoning tasks such as understanding geospatial intelligence data (Lebiere et al. 2013; Paik et al. 2013). Spatial abilities, including the ability to store spatial properties of an object such as location, size, shape and orientation, were added to Soar with the Spatial-Visual Imagery (Soar+SVI) module (Lathrop and Laird 2009). Soar has also been used to investigate spatio-temporal reasoning (Wintermute and Laird 2008).

In human studies, the ability to solve geometric analogy problems is the basis of Raven's Progressive Matrices (John Raven, John Carlyle Raven, and John Court 2004), a popular intelligence test. This task has been the subject of a number of investigations into computer spatial reasoning (Lovett, Andrew, Forbus, Kenneth, and Usher, Jeffrey 2010; Lovett, Forbus, and Usher 2007; Tomai, Forbus, and Usher 2004; Evans 1968; Evans 1964).

Most of the problems currently studied in spatial and temporal reasoning are, at their core, constraint satisfaction problems. Consider the simple task of determining whether an object C is in front of another object A given that A is in front of object B and B is in front of C (Lee and Wolter 2011). While it is possible to precisely measure the distance between A and B, the important information is the relationship between the objects. Given that domains involving time and space are frequently infinite in size, many spatial and temporal reasoning problems are intractable when the domain is modeled quantitatively. It is computationally more efficient to reason in terms of qualitative constraints (e.g., behind, below, intersecting, etc.) than quantitative measures (29 seconds after, 3° and 15cm distant, etc.) (Lücke and Mossakowski 2010; Mossakowski, Maeder, and Lüttich 2007). For this reason, most spatial and temporal reasoning research uses qualitative reasoning. Domains are modeled in terms of primitive relationships between abstract objects (Cohn and Renz 2008; Mossakowski, Maeder, and Lüttich 2007; Cabalar and Santos 2006). Other advantages of qualitative representations for spatial and temporal reasoning (goals and inputs are normally qualitative, easier for system users to understand and manipulate, etc.) are given in (Freksa 1992a).

In temporal reasoning, where there is only one dimension, domain elements are represented either as points (e.g., noon) or intervals of time (e.g., today) (Allen 1983; Ladkin and Maddux 1987; Van Beek and Cohen 1989; Vilain, van Beek, and Kautz 1989). The notion of intervals was introduced in Allen's seminal work on temporal calculus (Allen 1983). He listed 13 possible relationships between two items, *before*, *meets*, *overlaps*, *during*, *starts* (at the same time), *finishes* and the inverse of each of these (after, disconnected, not overlapping, etc.), plus *equals*. These relationships are referred to as *Jointly Exhaustive and Pairwise Disjoint* (JEPD) – they describe every possible configuration of (relationship between) two items and each relationship is mutually exclusive with the others (i.e., a given arrangement of objects cannot match two different relationships). Allen's use of intervals and the 13 relationships constitutes Allen's interval algebra, one type of temporal calculus. Others include Vilain's & Kautz's point algebra (Vilain, van Beek, and Kautz 1989), Freksa's semi-interval (Freksa 1992b) and Pujari's interval-and-duration (Pujari, Kumari, and Sattar 1999).

Spatial reasoning is similar to temporal reasoning in that domains can be represented as sets of primitive relationships between abstract objects but the larger number of dimensions in spatial domains requires more relationships and new representations. Rather than intervals, spatial calculi talk about regions (S. Li and Ying 2003a). In some instances, the specific shape of a region is important and thus there are spatial calculi that represent shape (Meathrel and Galton 2001). Other qualitative spatial representations include position (Clementini, Felice, and Hernández 1997), (qualitative) distance (Hernández, Clementini, and Felice 1995; Picazo-Tadeo, Reig-Martinez, and Hernandez-Sancho 2005) and orientation (Moratz, Dylla, and Frommberger 2005). As with qualitative temporal reasoning, spatial relationships are normally modeled as pair-wise qualitative relationships (Randell, Cui, and Cohn 1992; Cohn and Hazarika 2001), although there are domains that require the use of ternary relationships (discussed below) (Freksa 1992a).

Spatial calculi can be divided into two broad groups. *Topological calculi* focus on topological aspects of space such as shape, distance and position. In temporal calculus, relationships existed not just between pairs of items but between items and a reference point, specifically *forward*. This made it possible to say that one item came before or after another. Topological calculi do not track the relationship between an item and a reference point (up, north, front, etc.). Because of this, it is possible to say that one item is not connected to (i.e., inside of, intersecting or touching) a second item but not to say that an item is in front of, after or below it. In contrast, *directional calculi* consider direction to be an important part of the domain and assume some form of local or global reference system.

The most prominent topological spatial calculus is Randell's Region Connection Calculus (Randell, Cui, and Cohn 1992; Cohn and Hazarika 2001). Inspired by Allen's interval calculus for temporal reasoning (Allen 1983), Randell's RCC8 used regions rather than points and defined eight jointly exhaustive and pairwise disjoint relationships. The relationships are *equal*, *disconnected*, *partially overlapping*, *externally connected* (objects are touching), *tangential proper part* (objects are touching, the first object is inside the second), *tangential proper part inverse* (objects are touching, the second object is inside of the first), *non-tangential proper part* (the first object is inside the second) and *non-tangential proper part inverse* (the second object is inside the first). Because there is no frame of reference from which to determine direction, RCC8 has no equivalent to Allen's *before*, *after*, *starts with*, *started by*, *finishes with* or *finished by*.

Numerous extensions to RCC8 have been proposed to deal with moving objects and similar environments where objects rarely touch without intersecting (RCC-5) (Bennett 1994; Jonsson and Drakengren 1997; Thau, Bowers, and Ludäscher 2008), convex hulls (Cohn, Randell, and Cui 1994, 94), concave regions (Renz 2002a), contact algebra (Dimov and Vakarelov 2006) and n-dimensional domains (Renz 2002a). Developing efficient methods for detecting system-level constraint violations (as opposed to pairwise violations) is considered one of the fundamental tasks in qualitative spatial reasoning (Bennett 1998; Renz 2002b; Renz and Nebel 2007). Researchers have looked at the issue from the point of view of mathematical foundations (Bodirsky and Wölfl 2011; J. J. Li, Huang, and Renz 2009; J. J. Li, Renz, and Sanjiang 2008; Renz and Ligozat 2005; S. Li and Ying 2003b; S. Li and Ying 2003a), computational complexity (Bennett 1998; Renz 2002b; Renz and Nebel 2007) and approximation of human spatial reasoning (Renz, Rauh, and Knauff 2000; Ragni, Tseden, and Knauff 2007).

There are many well-known directional spatial calculi. Whether the directions are based on absolute (e.g., north) or local (e.g., to my left) frames determines whether they are orientation or relative orientation calculi, respectively.

An *orientation calculus* uses a global frame of reference. The best known orientation calculus is Frank's Cardinal Directions Algebra (CDA) (Frank 1991). Objects are represented as points and relationships are binary relationships based on North, South, East and West that allow one to say that Austin is Northwest of Houston and Denver is Southeast of Seattle.

The earliest work on relative orientation calculi was Freksa's semi-interval calculus for temporal reasoning (Freksa 1992b), based on Allen's interval calculus (Allen 1983). Using a cognitive analogy, Freksa argued that in the real world, we rarely know the duration of every event (in Freksa's example, we might know when someone was born or died but not necessarily both). This lack of information forces representations to be more complex (via disjunctions of *could be* relationships) which makes inference less efficient. Freksa argued that one could draw meaningful inferences (e.g., Newton lived before Einstein) knowing one end point of a given duration (Newton's death was before Einstein's birth; knowing Newton's date of birth does not help this inference). Freksa's solution was to create the notion of a *conceptual neighborhood*, where two relationships are *conceptual neighbors* if one can move *directly* from one to the other by moving one of a semi-interval's end points, referred to as a *semi-interval*. As an example, by extending interval X's end point, the relationship *before(x,y)* can change to *meets(x,y)* without going through any intermediate relationships. In contrast, *before(x,y)* cannot turn into *overlaps(x,y)* without moving through *meets(x,y)*. Conceptual neighbors form a conceptual neighborhood and a conceptual neighborhood replaces a set of disjunctions, making representation and reasoning more efficient. Between two one-dimensional intervals, a new set of 11 jointly exhaustive and pairwise disjoint relationships was created to allow for reasoning with incomplete knowledge (referred to as *coarse-grained knowledge*). These relationships are *older*, *younger*, *head to head*, *tail to tail*, *survives*, *survived by*, *precedes*, *succeeds*, *contemporary of*, *born before death of* and *died after birth*.

The notion of intervals and conceptual neighborhoods was extended to spatial domains in (Freksa 1992a) to allow reasoning in two dimensions with coarse-grained (imprecise, partial, subjective) data. Reasoning was done from the perspective of an agent in a given environment. An agent in the real world would not have a full view of the map or their position as a coordinate on a grid. The agent would merely know where it was relative to other, visible objects. Using Single Cross Calculus (SCC), an agent that knows where it and another object are relative to a shared reference point can determine the relative location of the object to the agent. A directed line is drawn from the origin point A (where the agent is) to a reference point B, indicating that A is facing (oriented to) B. This line separates left and right, relative to an object at A facing B. A second, perpendicular line is drawn through B, separating front and back. The combination of left-right, front-back is used to divide the area around B into eight regions: *straight-front*, *right-front*, *right-neutral*, *right-back*, *straight-back*, *left-back*, *left-neutral* and *left-front*. This reference frame is used to determine where a third point C is in relation to B. This forms the ternary relationship (A, B rel C) describing the location of C relative to B as seen by A. More complicated relationships can be represented using two single cross relationships, referred to as Double Cross Calculus (DCC) (Freksa 1992a; Scivos and Nebel 2001).

Single and Double Cross Calculus are sometimes referred to as coarse-grained Relative Orientation Algebra (ROA). Relative orientation, stating where an object is relative to a second as seen by a third and represented by a ternary predicate, is used in a number of other spatial calculi. Other well known relative orientation calculi include Dipole Calculus (Schlieder 1995), Dipole Relation Algebras (Dylla and Moratz 2005; Moratz, Renz, and Wolter 2000; Gottfried 2004), Ligozat 's Flip Flop Calculus (FFC) and Scivos' LR Refinement to Flip Flop Calculus. OPRA_m (Moratz, Dylla, and Frommberger 2005; Moratz 2006) and the refinement OPRA*_m (Dylla 2008) represent the relative orientation of two points to each other without reference to a third point. Egenhofer's 4- and 9-intersection calculus (Egenhofer and Franzosa 1991; Egenhofer 1994) describes the relative orientation of polygons rather than points. A metric calculus for describing the relative orientation of two polygons is given in (Peuquet and Ci-Xiang 1987).

So far we have discussed calculi that are either metric or qualitative, spatial or temporal and, when oriented, using an absolute or relative frame of reference. Several researchers have looked at ways of merging different schemes.

The cCOA calculus attempted (with partial success) to integrate orientation calculus (Frank's Cardinal Directions Algebra) with relative orientation calculus (Freksa's coarse-grained Relative Orientation Algebra (ROA)) (Isli 2004). The ALCRP^3(D) description logic integrates topological calculi with relative orientation (Kaplunova, Haarslev, and Möller 2002). Systems that use both metric and qualitative measurements include Time Map (Dean and McDermott 1987), Universal Temporal Language (Dechter, Meiri, and Pearl 1991; Kautz and Ladkin 1991), SOAR/SRS (Wintermute and Laird 2008) and Soar+SVI (Lathrop and Laird 2009). SOAR/SRS also combines spatial and temporal reasoning. Soar+SVI uses integrated spatio-temporal reasoning, as discussed next.

Integrated spatio-temporal reasoning is not as simple as combining a spatial reasoning and temporal reasoning system. A simple but important spatio-temporal reasoning task is crossing a busy road (Wintermute and Laird 2008). It is not enough to know how wide the road is, how long it will take to cross it, how far away a car is or how fast the car is moving. The reasoner must be able to tell whether there is any point in time and space where the person and car will be in the same place at the same time.

Although there are many proposed research areas, it has been argued that relatively little work has integrated spatial and temporal reasoning (Gerevini and Nebel 2002; Kreutzmann and Wolter 2011). Based on a belief that qualitative spatio-temporal reasoning is primarily about physics, Kreutzmann proposed as a benchmark problem trying to bounce a ball into a potentially hidden area (Kreutzmann and Wolter 2011). Another proposed benchmark where space, time and physics interact closely is computer billiards (Smith 2006). Physics also plays a role in road design, where turns in the road of a certain degree of sharpness will cause cars at those locations to slow temporarily, affecting the position and speed of a car at a specified point in time (Lathrop and Laird 2009).

The connection to physics is less clear in other domains. Proposed areas of study involving physical bodies include robot navigation in crowded environments with moving obstacles (e.g., people) and crowded with moving people and scheduling trains on a shared track so that their "intervals" do not "overlap."

2.1.2 Psychology

The psychological literature on human spatial ability covers a number of distinct capabilities.

The term *spatial visualization* is often used to refer to skills related to imagining or recognizing objects after they have been moved or deformed. The most common test in this category, the Vandenberg Mental Rotations Test (MRT) (Vandenberg and Kuse 1978), requires the subject to recognize abstract 3D objects after they have been rotated in 3D space. The Guilford–Zimmerman Spatial Visualization Task (Guilford and Zimmerman 1948; Guilford 1956) shows the subject a rotated clock and asks them which numbers the hands will point to if rotated by a certain number of degrees. Piaget's water-level task (Piaget, Elkind, and Tenzer 1967) requires the subject to visualize the water line in a bottle after the bottle has been rotated. The Ekstrom paper folding test (Ekstrom et al. 1976) shows a piece of paper being folded. A hole is punched in it and the subject must predict where the hole will be when the piece of paper is unfolded. The Surface Development Test (SDT) (Ekstrom, French, Harman, & Dermen, 1976) is similar to building models from a sheet of paper. The edges of an arbitrarily shaped piece of paper are labeled and the subject must determine which edges will be touching after the piece of paper is folded into a 3D object.

Spatial relations, *spatial orientation* and *perspective taking* refer to tasks involving maintaining one's orientation with respect to other objects and making judgments about how they change. Perhaps the most common spatial relations test is the card rotation test (French, Ekstrom, and Price 1963), which requires subjects to recognize an image of an object after a planar rotation (i.e., rotated while facing the subject). The cube rotation test (Amthauer 1973) shows two cubes with letters and numbers on each side and asks the subject to determine if they could be the same cube. The Guilford–Zimmerman Spatial Orientation Test (Guilford and Zimmerman 1948; Guilford 1956) gives the subject two pictures of a scene as seen from a boat and asks them to determine how the boat moved to get from the first view to the second. Similar to Freksa's single cross calculus in artificial intelligence, the object perspective task (Kozhevnikov and Hegarty 2001) shows the subject a piece of paper containing several objects, asks the subject to imagine themselves as being at one of the objects and facing a second and then to indicate, from that perspective, the relative position of a third object. A newer category of tests, *dynamic spatial reasoning* (Pellegrino and Hunt 1989; Colom et al. 2002), involves predicting where a moving object is going and when it will arrive there.

In *route learning* tasks, the subject learns, from a first person perspective, the layout of an area and then must find a route to a given destination. Common tasks include finding one's way out of a maze using local cues such as room shape and hall length (Moffat, Hampson, and Hatzipantelis 1998) and locating an invisible object using only distant landmarks (Morris 1981). In *map learning* tasks, the subject is shown a map and later asked questions about it. Questions include giving directions, redrawing the map, pointing to locations, recalling landmarks and estimating which of two places is closer to a given location (Galea and Kimura 1993).

Several research areas involve low-level spatial abilities. We mention them here because, as we will discuss later in this section, they play a role in explaining higher-level task performance. *Line orientation* tests measure the ability to notice small differences in line orientation (A. M. Treisman and Gelade 1980; A. Treisman and Gormican 1988). *Visual search* tests measure a person's ability to find an object in a scene (Wolfe 1994). *Perceptual disembedding* tests measure a person's ability to find an object that blends into its surroundings (e.g., finding common geometric shapes in a larger design) (Oltman et al. 1971). A person's *useful field of view* is the area the person can extract information from with a brief glance without moving their eyes or head. The ability to extract information from this area requires several resources and abilities. The useful field of view test (Ball et al. 1988) measures a person's spatial attention (the amount of information they can track at one time), object detection, multiple object tracking and selective spatial attention (i.e., the ability to ignore irrelevant information).

Unlike with computers, there is no easy way to examine in any functionally usable way the internal representations of the brain (not that people haven't tried, for example

(Arentze, Dellaert, and Timmermans 2008)). We can, however, make inferences based on test performance. What can these spatial ability tests tell us about how humans represent space? The answers, unfortunately, are difficult to interpret.

The literature notes that many factors influence performance, for example, handedness (left handers do worse on spatial reasoning tasks) (McKeever et al. 1987). In studies of college students, subject major correlated with performance, with scientists outperforming those in social science and liberal arts (Peters et al. 1995). Many concepts of space are genetically coded but some facet of spatial reasoning performance, whether representations, reasoning or something else, must be learnable as performance on spatial tasks improves significantly with practice (Terlecki, Newcombe, and Little 2008).

Aside from practice affects, the largest determinant of spatial reasoning task performance is gender. Spatial reasoning is a sexually dimorphic ability, with men performing significantly better than women. In one multifactor study of psychological functioning, only 20-25% of women reached the average male performance on spatial reasoning tasks (Harris 1978). On a mental rotation task, gender explained on average 20% of the variance in scores (Peters et al. 1995). In a route finding task, men scored 1½ standard deviations above women for both time required and spatial memory errors (Moffat, Hampson, and Hatzipantelis 1998). In a test of mental rotation that factored out the effects of age and general intelligence, men scored a full standard deviation higher than women (Johnson and Bouchard Jr 2007).

The male advantage in spatial reasoning is not merely large, it is pervasive; "sex difference favoring males in spatial performance is one of the most reliable of all cognitive sex differences in humans" (Moffat, Hampson, and Hatzipantelis 1998). In spatial visualization and orientation, men have significantly outperformed women in 2D mental rotation (D. W. Collins and Kimura 1997; Sanders, Soares, and D'Aquila 1982), 3D mental rotation (Linn and Petersen 1985; Voyer, Voyer, and Bryden 1995), the

Guilford–Zimmerman clock task (Linn and Petersen 1985), Piaget's water-level task (Astur, Ortiz, and Sutherland 1998) and the Ekstrom paper folding test (Linn and Petersen 1985). In route learning, men significantly outperformed women in navigating with (Moffat, Hampson, and Hatzipantelis 1998) and without (Astur, Ortiz, and Sutherland 1998) local cues. In map learning, men outperformed women in determining direction (Holding and Holding 1989), giving directions without errors (Miller and Santoni 1986), giving directions quickly (Moffat, Hampson, and Hatzipantelis 1998), reporting cardinal direction (Ward, Newcombe, and Overton 1986), reporting distance (Miller and Santoni 1986) and integrating overlapping maps into a larger map (Galea and Kimura 1993). In an analysis of data from 42 mental ability tests where the effects of general intelligence and age had been removed, men significantly outperformed women in every test of spatial visualization (Johnson and Bouchard Jr 2007).

Many attempts have been made to explain these differences developmentally or in terms of test bias without much success. There is little or no support for a decreasing gap over the decades (Masters and Sanders 1993; Voyer, Voyer, and Bryden 1995). In spatial visualization and orientation tasks, the gender gap remained even when subjects were told to imagine themselves rotating rather than the object (Kozhevnikov and Hegarty 2001), using simple planar rotations (D. W. Collins and Kimura 1997), in dynamic environments (Contreras et al. 2001), with practice (Terlecki, Newcombe, and Little 2008; Baenninger and Newcombe 1989; Marulis et al. 2007; Spence et al. 2009), using significantly longer training periods (Terlecki, Newcombe, and Little 2008) and accounting for experience with first person shooter video games (Marulis et al. 2007). In map and route learning tasks, the gender gap remained even when subjects performed route finding tasks on more realistic (first person rather than top down) tasks (Moffat, Hampson, and Hatzipantelis 1998), and accounting for experience with first person shooter video games (Moffat, Hampson, and Hatzipantelis 1998; Astur, Ortiz, and Sutherland 1998).

Much of the evidence for male superiority in spatial reasoning points to biology. By the age of three, before strong cultural conditioning takes hold, males are already superior to females at mental rotation (Kimura 1992). Hormones, particularly androgens (male hormones) such as testosterone, strongly impact spatial reasoning performance. Animal studies and studies with children born with congenital adrenal hyperplasia, a disease that affects androgen levels, have shown that higher androgen levels in women lead not only to stereotypical male behavior (Berenbaum and Hines 1992; Berenbaum 1999) but male-level spatial performance (Gouchie and Kimura 1991). A person's androgen level during early childhood, when the brain is still being wired, strongly affects spatial ability while changes in androgen levels after development do not (McKeever et al. 1987; Kimura 1992), suggesting that the basis of spatial performance is hard wired during childhood, although small affects can be seen by varying androgen levels in adults caused by menstruation (Hausmann et al. 2000), aging (Cherrier et al. 2001) and seasonal fluctuations (Kimura and Hampson 1994) but not daily fluctuations (Silverman et al. 1999). An evolutionary account for why this might be, based on male ranging behavior in hunter gatherer society, is given in (Gaulin and FitzGerald 1986; Silverman and Eals 1992; Sherry and Hampson 1997).

Given that men and women show roughly equal general intelligence (Johnson and Bouchard Jr 2007), if men outperform women in spatial reasoning tasks, there is presumably a difference between the two that can help us understand factors affecting spatial performance. These differences might be in the resources available or how those resources are used.

General intelligence and age play a large role in task performance, masking the influence of gender. When these factors were removed from a study of 42 mental ability tests (Johnson and Bouchard Jr 2007), women outperformed men in nine tests: vocabulary, spelling, word fluency, word beginnings/endings, meaningful memory, associative memory, pedigrees (identifying relationships in a family tree), coding (identifying symbol-number pairings) and perceptual speed. In general, women scored higher in measures of memory and verbal ability. This same study identified two negatively correlated abilities that they argued formed tradeoff dimensions. The *rotation-verbal* dimension implies that increased mental rotation ability requires lowered verbal ability and vice versa, consistent with findings that men are, on average, better at mental rotation and women are better at verbal skills. This is supported by research on transsexuals where male-to-female transsexuals showed increased verbal ability and female-to-male transsexuals showed increased mental rotation ability (Slabbekoorn et al. 1999). The *focus-diffusion of attention* dimension measures how attention is allocated. Men were more likely to focus attention on a single item while women were more likely to spread their attention across multiple items. Other studies have shown that men are more field-independent in perceptual processing (i.e., focusing on an item independent of its context) while women were more field-dependent (Kogan and Dorros 1978). This likely explains why women do worse on the perceptual disembedding spatial task (e.g., finding a shape embedded inside a more complex shape) (Oltman et al. 1971).

Women have the advantage in object memory, verbal skills and perceptual speed. They are also less likely to hyper-focus on a single item and more likely to integrate context and background information. This is presumed to affect the methods women use to reason about space. In tests of map and route learning, men talk more about geometric properties (e.g., angles), quantitative measures (e.g., distance) and absolute frames of reference (e.g., north) while women were more likely to talk about landmarks, switching to geometric representations only when there were few landmarks (Chai and Jacobs 2009; Galea and Kimura 1993; Grön et al. 2000; Lawton 1994; Levy, Astur, and Frick 2005). Women outperformed men in recalling landmarks and street names, although this has been shown to be at least in part due to reasons other than memory. Unfortunately, higher scores on landmark recall correlated with lower scores on all other map learning tasks (Galea and Kimura 1993).

One interesting and under-explored avenue of research concerns selective spatial attention. Selective spatial attention is the ability to allocate attention primarily to important objects rather than dividing it between important and distractor objects. Studies of useful field of vision have found that women perform below men in measures of selective spatial attention (Feng, Spence, and Pratt 2007), consistent with findings that women spread their attention more widely than men (Johnson and Bouchard Jr 2007). It has been argued that selective spatial attention is important to abilities of mental rotation, training can eliminate the gender difference in selective spatial attention and correspondingly decrease (but not eliminate) the gender gap in mental rotation (Feng, Spence, and Pratt 2007).

The mechanisms of selective spatial attention are of interest to the work described here. Perception takes place in two stages (Duncan 1984). The first stage segments visual imagery into a set of objects. It is done unconsciously, in parallel and before attention is allocated. In the second stage, focal attention (attention under conscious control) is allocated to a single object, allowing additional detail to be gathered. This process is parallel in properties of the object and serial across objects – the viewer can make multiple, simultaneous judgments about the attended object but cannot make judgments about two or more objects at the same time.

As will be argued later in this paper, the effectiveness of problem solving depends strongly on how the components of the problem are represented. Specifically, we argue that efficient reasoning relies on representing objects in terms of the actions they afford. The first stage of perceptual analysis determines which objects are perceived in the environment. Stimuli are divided into objects based on Gestalt (object-level) properties such as spatial proximity, color, movement and continuity of contour (Duncan 1984). This process is also affected by the problem solver's experience with the problem solving context, dividing a scene into objects (including higher level concepts) based on the effectiveness of those representations in prior problem solving instances (Goldstone and

Barsalou 1998). From the resulting set of objects, attention is allocated based on the problem solver's goals which, in turn activates relevant problem solving strategies (Houghton and Tipper 1994; Marr 2003).

Most of the discussion thus far has been on spatial representations. What of time?

Spatial and temporal reasoning systems in artificial intelligence often represent their domains as a set of constraints. When treated this way, there is little difference between space and time (Valdes-Perez 1986).

For humans, space and time are not interchangeable but they are related. People often use spatial terms to describe time. Duration can be described as having had a *long* vacation or gone to a *short* concert. Ordering can be described as looking *forward* to the weekend and putting the past *behind* us. Time can be a place, such as when we talk about *back* in the good old days and the future that is *ahead* of us. (Casasanto and Boroditsky 2008; Fuhrman and Boroditsky 2010; Fuhrman et al. 2011; Miles et al. 2010). People are far less likely to use temporal terms when discussing space (Casasanto and Boroditsky 2008).

Why might this be? The Metaphoric Structuring theory (Boroditsky 2000; Casasanto and Boroditsky 2008) says that we understand things we cannot directly experience through metaphorical mappings to things we can. Specifically, when we reason about time, an abstract concept, we create a mapping from time to space, a concrete domain we have physically experienced, and reason from that. The Perceptual Meaning Analysis theory (Jean Matter Mandler 2004; Jean M. Mandler 2012) argues that, during their first few months of life, infants interact with, attend to and learn about space and that the resulting spatial concepts form the primitives used for concept formation and simple problem solving. Infants learn a small set of path, motion and spatial relation primitives to interpret spatio-temporal information. Abstract concepts take longer for children to learn because of their distance from experiential (perceptual and motor) data. While young children can reason about time, it has been argued that they cannot reliably distinguish between space and time until the age of nine, with temporal deductions being based primarily on spatial information (Piaget 1927).

Linguistic data shows that people often use spatial metaphors to discuss time. Until recently, relatively little work has been done to determine whether this type of asymmetric spatio-temporal conflation affects anything more than how we talk about time. Barsalou argues for embodied temporal processing, where temporal reasoning is grounded in the sensory-motor system (Barsalou 2008). If true, space should affect non-verbal aspects of temporal reasoning. In a series of studies by (Casasanto and Boroditsky 2008), when asked to estimate either how long a line was (length) or how long it was shown on the screen (duration), estimates of length were highly accurate and unaffected by duration but estimates of duration were affected by line length, with longer lines being reported as having longer durations. In a time-classification task (a subject must select one of two options on opposite sides of a screen by moving a mouse, with the time and movement profile of the mouse recorded), (Miles et al. 2010) showed that, for English speakers, words related to the past caused the motor system to bias to the left while words associated with the future biased it to the right. Although there was a strong correlation between the direction of time (past and future) and motor activation (left and right), other studies have shown that the mapping between the two depends on the subject's primary language. For English speakers, who read left to right, the past biased actions to the left while for Hebrew speakers, who read right to left, actions were biased to the right.

Language and culture have been shown to affect how people talk about time, with English speakers using a left-to-right orientation, Hebrew and Arabic speakers using right-to-left and Mandarin Chinese speakers using top-to-bottom (Fuhrman and Boroditsky 2010; Fuhrman et al. 2011). These linguistic metaphors spill over into how people sketch temporal relationships (Fuhrman et al. 2011) and place objects based on temporal relationships (B. Tversky, Kugelmass, and Winter 1991). The language a person speaks also affects how people think about time, including whether time is stationary or moving and whether it is limited or open-ended (Fuhrman et al. 2011). In tests of people who were bilingual in English and Mandarin Chinese, how subjects conceptualized time depended on whether they were tested in English or Mandarin (Fuhrman et al. 2011).

While time is not space, it has been argued that they do have a similar conceptual structure and that using spatial information to reason about time is just as useful as using temporal information (Boroditsky 2000). The impact of using spatial information and representations to solve temporal problems in tower defense puzzles, a type of complex spatio-temporal puzzle, was tested in (Wetzel et al. 2012). Analysis showed that conflating time and space achieved good performance but precluded the use of problem solving strategies keyed to affordances that only exist in temporal representations.

2.2 Problem Solving for Complex Tasks

Spatial and temporal reasoning in artificial intelligence has primarily focused on representing problems as sets of qualitative constraints (Cabalar and Santos 2006; Cabalar and Santos 2011; Lee and Wolter 2011; Lücke and Mossakowski 2010; Mossakowski, Maeder, and Lüttich 2007) and solving them using a path consistency algorithm (Aho, Hopcroft, and Ullman 1974; Allen 1983; Beek and Manchak 1996; Lee and Wolter 2011; Mackworth 1977; Montanari 1974; Renz and Ligozat 2005; Renz and Nebel 2007). Much of the debate for problem solving has been on finding more efficient ways of solving systems of constraints, with recent calls to use methods from outside computer science such as the simplex method from linear algebra, Wu's method from elemental geometry, the Gröbner base method from algebraic geometry and the cylindrical algebraic decomposition method from real algebraic geometry (Lee and Wolter 2011). The precise methods for efficiently solving constraint satisfaction problems are well documented in the literature and independent of the approach taken in this work and so we end our discussion of them here.

At the start of this work we made the assumption that techniques for solving complex spatio-temporal reasoning problems would be general enough to work on a broad spectrum of complex problems. The success of current spatial and temporal reasoning methods in artificial intelligence has been in part due to a focus on the problems themselves and has led to the use and development of methods that are specific enough to a certain type of problem that they do not generalize well to other types of problems. The tower defense problem discussed in this work, for example, seeks to find an optimal configuration of items as opposed to a legal set and thus is not amenable to constraint satisfaction approaches. At the same time, we hoped to discover methods that could exploit, in a general way, information in the environment and were necessarily more specific than traditional AI problem solving approaches such as search, logical inference and policy learning.

The approach taken by this work is to study how humans solve complex problems. Unfortunately, there is no consensus in the psychological literature on the definition of a complex problem. Gonzalez (Gonzalez, Vanyukov, and Martin 2005) argues that there are at least four categories of complex problems and their general approach of study: dynamic decision making (Brehmer 1992; Edwards 1962), complex problem solving (Frensch and Funke 1995), systems thinking (Sweeney and Sterman 2000; Senge 1997) and naturalistic decision making (Klein 1998; Lipshitz et al. 2001). Although the specific rationale for classifying a problem into one of these categories was not given, every category had one thing in common – each involved understanding and then controlling dynamic, complex situations. More specifically, complex problem solving took place in situations containing hidden processes and non-linear relationships where the goal of the problem solver was to learn the causal relationships between the system variables and use this knowledge to control the system (Goode and Beckmann 2010).

By this definition, tower defense would not qualify as a complex problem. The tower defense system studied here is deterministic and perfect information– the problem solver has access to all relevant information. At issue is whether the problem solver is looking at a given situation in the right way. Gaschler argues that skill is based significantly on information reduction – with practice, the problem solver learns which information is important and which can be ignored, allowing the problem solver to use cognitive resources more efficiently (Gaschler and Frensch 2007). This, too, differs from what we believe is happening in the type of complex problem solving studied here. In the tower defense task, time is not an issue – the problem solver has as long as they need to create a solution. The problem solver has time to evaluate all given information, relevant and not. If the key to solving these problems was to uncover hidden information and have sufficient time to consider all important information, we would expect all problem solvers to perform the same on tasks such as tower defense where those factors are removed. As shown in later chapters, this is not the case.

In her analysis of complex problem solving approaches, Gonzalez (Gonzalez, Vanyukov, and Martin 2005) argues that there are three broad categories of research environments. Traditional psychological experiments are performed in a laboratory with very simple tasks. The researcher has precise control over experimental conditions which they can use to focus the work on specific aspects of behavior but loses the context of real world problem solving and the impacts that context might have on behavior. In naturalistic decision making, researchers study people in the field, interviewing and observing people engaged in real world behavior. This allows the researcher to see behavior under real world conditions but removes much of their ability to control the experiment. The third and newest approach is to conduct studies in complex microworlds, also known as synthetic task environments, high fidelity simulations, interactive learning environments, virtual environments and scaled worlds. These computer-based simulations can offer many (although certainly not all) of the factors involved in real world problem solving while allowing the experimenter to control the experiment and remove or abstract out parts of the environment that might act as distractors.

An analysis of popular microworlds used in psychological research showed that many derived their complexity from time pressures, non-linear relationships between variables and hidden variables and processes (Gonzalez, Vanyukov, and Martin 2005). The research described in this paper was conducted in GopherTD, a microworld we created to investigate a problem known to be challenging yet tractable for humans but whose complexity was not due to hidden variables or information overload. It is important to note that the GopherTD microworld is not merely an approximation of a real world task but an actual real world task in itself. It is a research version of a video game played by millions of people across the world. One of the benefits of using popular commercial video games as research tools is that they contain all of the elements a real world task (i.e., playing the game) has.

Rather than trying to uncover hidden information or trying to find the relevant subset of information in the information available, we believe that solving the types of problems exemplified by tower defense requires re-representing the information presented into abstract yet actionable concepts. More specifically, we believe that problem solving is accomplished by an actor setting a goal, representing the items in their environments in terms of the opportunities they provide and selecting simple, focused strategies keyed to those affordances. Consider the problem of determining which of two fractions is larger. In the United States, students are taught to find the least common denominator, convert both fractions to the same denominator and then compare the numerators. In practice, students rarely use this strategy (Behr et al. 1984; Smith, John P., Disessa and Roschelle 1993). Specific instances of the problem might contain properties that afford simpler strategies. One such strategy is the half-way point strategy. If students can quickly determine that one fraction is less than 1/2 and the other is larger, the fraction greater than half is the larger fraction. Similar strategies existed for the reference points 1/4, 1/3 and 3/4.

Although the common denominator strategy works for all fractions, strategies such as the half-point strategy require fewer calculations and less memory. These strategies are sometimes referred to as fast and frugal heuristics because they are quick to execute, require minimal memory and work with incomplete information (Bertel and Kirlik 2010; Reimer and Rieskamp 2007). This is sometimes referred to as ecologically rational decision making because these properties are important for solving the types of problems humans are likely to encounter in the real world, where decisions must be made with limited time and knowledge (Gigerenzer and Goldstein 1996). By exploiting affordances in a problem, strategies gain the ability to be fast and frugal at the expense of generality. Studies of expertise have shown that both novices and experts prefer to use an array of simple, special-purpose strategies rather than a single complex, general-purpose one (Smith, John P., Disessa and Roschelle 1993). Although these strategies seek to be ecologically rational rather than logically consistent (i.e., they focus on specific problems and immediate goals rather than trying to create a comprehensive model of how a system works) (Reimer and Rieskamp 2007), they have been shown in experiments to outperform complex methods from artificial intelligence and machine learning (Bertel and Kirlik 2010; Gigerenzer 2004; Gigerenzer and Todd 1999b; Reimer and Rieskamp 2007).

How well a person can reason about a problem depends on how the problem is represented. For example, in studies of both doctors and elementary school children, problem solvers were significantly more accurate at solving probability problems when the facts were phrased in terms of how humans would encounter the data in the real world (i.e., natural frequencies) rather than probability (Gigerenzer 1991; Gigerenzer et al. 2008; Ruscio 2003; Sedlmeier and Gigerenzer 2001). The representation is important because the representation can do part of the computational work (Zhu and Gigerenzer 2006). Examples of this are shown later in this work.

How a problem is represented also affects the strategies chosen to solve it. Strategies are closely tied to the affordances of the environment. Affordances were first described by Gibson as properties in the environment relevant to the problem solver's goals (Gibson 1979). These are not merely passive constructs; rather, they actively suggest actions for the agent to take. For example, a pull handle on a door encourages pulling while the handle of a coffee mug encourages grasping (Norman 2002). This is not a result of a person actively examining their environment, attempting to identify relevant properties and considering what actions are available, it is an automatic process guided in part by the perceptual and motor systems. When objects that can be acted on are seen, the corresponding motor systems activate (Thill et al. 2013; Tucker and Ellis 2004).

Representations of objects are believed to be functionally encoded, that is, objects are, in part, represented by the actions we can take on or with them and the properties affecting how we use them (e.g., size, weight) (Gorniak and Roy 2007; Thill et al. 2013). Likelihood of using them also plays a role. For example, subjects made judgments about objects more quickly if they were described in terms of functional verbs (e.g., write with the pen) or manipulation verbs (e.g., move the pen) than by observational verbs (e.g., look at the pen) and also were faster when the object was within peripersonal space (i.e., they could reach it) rather than out of reach (Costantini et al. 2011).

This has led to the argument that strategy use, and by extension reasoning in general, is at least in part, perceptually driven (Marr 2003). Experience, however, affects perception. By monitoring the outcomes of applying strategies, new affordances can be learned. This, in turn, causes people to look at problems differently (Goldstone and Barsalou 1998). Because of the link between affordances and strategies, it can therefore be argued that the strategies one uses can affect perception (Canas et al. 2003).

Much of the research on strategies has looked at simple problems that can be solved with a single strategy. The work described in this paper looks at complex problems where, we argue, the problem solver must use several strategies concurrently. We present a catalog of strategies applicable to problems in the tower defense domain. Similar strategy catalogs have been presented in the artificial intelligence literature for football (G. Collins 1987). As we do in this thesis, these strategies can be analyzed to discover representations necessary to support the strategies. The number of strategies and accompanying concepts needed to solve complex problems can number into the hundreds. In the most comprehensive project of this type, Gordon cataloged 372 pre-formal (explicitly but not computationally defined) strategies from 10 domains (business, politics, war, music, etc.) and derived from that roughly 1,000 concepts organized into 48 conceptual groups (Gordon 2004a; Gordon 2004b). The idea of transferring explicit strategies from classical domains to microworlds for study is proposed in (Gordon 2001).

Chapter 3

Background

3.1 Desired Properties of a Domain

The work described here covers many different but related topics. At a broad level, we are looking at how humans solve complex problems with significant spatio-temporal components and how this might help us to design computer programs to do the same.

All work must start somewhere so this work began in a location most humans will be familiar with – objects moving through space over time. More specifically, how do we solve problems that have:

- **Predominantly Spatial Actions**. The problem requires the problem solver to take an action related to space. Such actions include aiming, placing and moving. As discussed below, this research focused on placement decisions. Such decisions could include deciding where to seat guests at a wedding, place cars in a parking lot, locate restaurants in a mall or setup cameras in a bank.

- **Non-Spatial Goals**. While the actions are spatial, the goals are not. For example, one might concentrate restaurants in a mall into a food court (a spatial action) in order to increase sales (a non-spatial goal).
- **Meaningful Space-Time Interaction**. Much work has been done on representations and reasoning for space and time as individual quantities but our interest is in a specific type of interaction between the two - objects moving through space over time where time and space interact and where time plays an important role. We are interested in domains where time is used for more than sequencing (e.g., not simply step A comes before step B). In the search and rescue domain both space (the location of buildings, fires, etc.) and time (time to reach a building, time to put out a fire) are important and must be reasoned about jointly rather than independently. A large part of solving such problems is predicting the state of the system at given points in time.

If we wish to make computer programs that can work in the real world, we do not need to study real-world problems but we must study problems that have real-world properties. For the work discussed here, the properties of interest are:

- **Complexity**. Very good studies have been done on how humans solve simple problems. Much of this work was done on tasks that involved a single decision (comparing fractions, estimating city size, etc.). It is our belief that complex problems are not solved in the same way as small problems but are, instead, approached in a qualitatively different way. To test this belief, we looked for problems with higher complexity.
- **Definition**. It is our belief that problem representation plays a big role in how difficult it is to solve a given problem and that much of human intelligence is in feature selection and feature creation. In the real world, there are countless variables, some valuable, many not, so one task is determining which variables to use in problem solving. Many of the important variables are not in the raw data but at a higher level. Many of these meta-features are relationships between

variables. A second task in problem solving is creating useful features from the raw data. For us to test the impact of representation, we needed a domain that contained numerous variables of differing value where problem solving was greatly assisted by the use of meta-features that the problem solver must create.

- **Variation**. In the real world, problem solvers rarely encounter the same problem twice. People constantly face problems that are similar enough to previous problems they've encountered to allow them to leverage their experience but different enough that they cannot use previous solutions.

Humans and computers often (although not always) work in different types of environments on different types of problems. They are optimized to their environments – computers are not humans and the two do not have the same strengths and weaknesses. It should not be assumed that studying humans will be of value to computer science, just as it should not be assumed that it cannot. If we are to use human data to improve computer programs, it makes sense to study areas where humans have something to offer to computer science. This research looked for domains that had the following properties:

- **Ecological Validity**. The theory of ecological validity says that evolution does not create life that succeeds in general, it creates life that succeeds in its environment. Dolphins are optimized for oceans, camels for deserts, birds for the air. Likewise, humans evolved in an environment where certain types of problems were common and human abilities were optimized to be successful there. If we wish to learn something from humans that will benefit computers, it makes sense to start with problems that humans are good at. We looked for domains that represented problems humans frequently or voluntarily encounter.
- **"Average Human" Success**. Some people are considered "geniuses" because they can do things few other people can do. For example, while chess geniuses are as good as the best computers, the average person has no facility for chess. This implies that evolution did not prepare people for domains such as chess. We chose to study domains where performance was good even for the average person.

- **Humans Significantly Outperform Computers**. Computers solve some problems, such as scheduling and optimal path finding in large data sets, much better than humans. Computers solve other problems, such as chess, as well as humans but in a significantly different way. Our interest is in problems where computers significantly under-perform humans.
- **Human Data is Meaningful to Computers**. If our goal is to study humans in order to learn how to better design software, the lessons we learn from human data must be implementable by computers. It is hard to pre-screen domains for this property but it is obviously a desirable one.

Many of the above requirements are tied to real-world properties and thus apply to many of the problems humans face in their lives. The more constraining requirements are that we are interested in problems with spatial actions, non-spatial goals and a significant time-space interaction. Examples of real-world tasks with these properties include:

- **Fixed Location Defense**. The military (whether real, simulated or in video games) places trenches, moats, fences, guard towers and soldiers at strategic locations to protect the base. For a prison, one approach to reducing the frequency, severity and duration of riots is careful placement of gates, corridors, cameras and guards. Bank robberies are managed by careful placement of the vault, routes to the vault and placement of cameras, gates and guards.
- **Crime Recovery**. The layout of a prison and the surrounding environment can assist in capturing escaped prisoners. In newer suburbs, city planners use curvy roads rather than a grid to slow traffic within a neighborhood and limit the number of entrances and exits from the neighborhood to make it easier for police to predict where criminals will emerge.
- **Threat Prediction**. For a sniper, there are certain properties (line of sight, seclusion, safe exit route, etc.) that make one location better than another. It is the responsibility of a security organization to predict these spots. When an important person is set to speak outside, guards predict where potential threats are and

respond accordingly. If the person is part of a parade, guards scout the area for the likeliest danger spots, route the parade away from these areas if possible and secure or monitor the locations if not.

- **Disaster Response**. As discussed earlier, given limited resources, disaster response teams must decide which emergencies to respond to first – which fires to put out, which areas to search for victims, where to place fire breaks, etc. Police attempting to control widespread riots and looting must decide where to place their resources to minimize the damage done.
- **Store Layout**. Planograms show where every store in a mall, department in a store, aisle in a department and product in an aisle are located. Considerable attention often goes into deciding which products are closest to the door, where sought after items are placed, where impulse purchase items are placed and how paths are laid out in a store. As an example, at one electronics retailer, the primary path through the store (called the racetrack) is circular and items people come in looking for (e.g., appliances) are placed at the back, causing these shoppers to pass through several departments containing impulse purchase items (CDs, movies, etc.) and increasing sales.

The problem with studying tasks that have real-world complexity is that studying them scientifically is also complex. We need to collect detailed data but it is hard to instrument the real world and studying people in a laboratory risks changing how they behave, especially in domains where the context that exists in the real-world tasks impacts how a person behaves. With humans, we lack each person's learning history – there is no way to know all of the experiences the person had before you studied them. In computer science, we can see not only a computer's behavior but how it was generated – we can read the source code to see why the computer made the decisions it did. Humans are, for the most part, a black box – we cannot read a human's "source code" to see why they made the decisions they did. We can speculate about how these decisions were made but for real-world tasks, with their complexity, context and imprecision, it is often hard to validate these speculations. There are no magical solutions to these issues but if we are careful about the domain we use, we can minimize them. The final criteria we used for selecting a domain were that it had the following properties:

- **Simple**. While we want the domain to be complex enough to study interesting behavior, it should be no more complex than necessary. Properties that are meaningless yet distracting should be minimized. As an example, if a problem can operate in continuous or discrete space and operating in continuous space offers no new insights, the discrete space should be preferred.
- **Focused**. We wish to attribute a problem solver's behavior to a specific goal. This becomes a problem if the problem has multiple, unrelated goals. Optimizing the placement of cameras in a sports arena while minimizing the cost is an acceptable set of related goals but trying to maximize a sports arena's security, concessions revenue, audience participation and parking is not.
- **Measurable**. There are many interesting behaviors that are not easily quantified or measured. This makes analysis of the data difficult. Our interest is in domains where the key properties, including behavior, can be precisely measured.
- **Configurable**. A simple, focused problem allows us to study the key variables in problem solving. As we develop hypotheses, we want to be able to test them so we want a domain where it is quick and easy to design new experiments that focus on just the items of interest.
- **Fast**. To get in-depth data, it is often necessary to have an experiment that requires a large time commitment from the people you are studying. Such experiments are often hard to schedule and can generate highly variable data, an outcome we experienced in our first experiment. It is important to select tasks that can be done in a reasonable amount of time, with shorter being better.
- **Motivation**. Because we are studying problem solving to learn useful behaviors, it is important that the problem solvers studied be motivated to perform well.
- **Previously Unknown**. Fluid intelligence represents how someone thinks about a novel problem and is used in solving novel problems. In order to study such "on-

the-spot" reasoning, we needed to use a problem that the problem solver had not seen before.

- **Easy to Capture Detailed, Meaningful Data**. We can only analyze information that we capture. It is important that we be able to capture the information we need with the resources we have available.
- **Multiple, Qualitatively-Different Instances**. As mentioned earlier, humans rarely encounter the same problem twice but often encounter novel problems in the same class. To properly replicate this, we need a domain in which it is possible to create problems that are similar enough to previous problems to allow the problem solver to leverage their experience but different enough that they cannot directly use previous solutions.

3.2 Domain: Tower Defense Puzzles

3.2.1 Tower Defense Puzzles

To measure spatio-temporal reasoning in humans we used a task called tower defense. It evolved from the base defense portion of real-time strategy games where the player tried to prevent enemies from reaching their base (castle, city, military headquarters, etc.) by placing guards (technically, guard towers) at key points along their path. More abstractly, the goal is to prevent a set of objects from reaching a location by placing a limited number of special-purpose towers at strategic locations along the path. A pictorial walkthrough of a game session is given in Section 3.3.1 GopherTD: Task.

Components. Tower defense puzzles have become a very popular genre of video games, with several hundred tower defense games having been made in the last decade. Each game is slightly different but all tower defense games have the following components:

- **Creeps**. The objects to be stopped are generically known as creeps since they creep along a path. Although often (but not always) representing enemies, the creeps make no decisions – they follow the path, never veering from it, never changing speed, never attempting to defend themselves. The creeps have a specified amount of health or similar resource that is lowered by the towers. When the creep's health is reduced to 0, it is removed from play.

- **Maps**. The path that the creeps follow is defined by the map. Different maps have different paths. Which towers should be used and where they should be placed depend on the map. Maps in tower defense games are often significantly dissimilar from each other. Each map is an independent problem with its own unique set of solutions.
- **Towers**. Towers "attack" the creeps that come into range. There are different types of towers. Most games have at least one tower that is fast, one that is strong and one that causes the creeps to temporarily move slowly. The player has no direct control over the tower; they choose which towers to use and where each tower is located but cannot tell it when to attack, who to attack, etc.
- **Money**. The problem solver is given a limited amount of money. Different types of towers have a different price.
- **Build Phase**. This is the time during which the player purchases and places towers on the map. When the player indicates they are done, the execute phase begins.
- **Execute Phase**. This is the time when the creeps enter, moving on the path, and the towers respond to them. The player cannot take any actions during this phase.

The specific tower defense game used for this research is called GopherTD. It is described in Section 3.3 Task and Tool: GopherTD.

Task. To prevent the creeps from reaching the protected location, the problem solver must select an appropriate set of towers and place them at appropriate locations. This requires the problem solver to analyze the map and perform spatial reasoning. When we began this research, it was not immediately apparent how important temporal reasoning was to this task. The importance of time is discussed in Chapter 6 Unifying Spatial and Temporal Reasoning.

3.2.2 Properties

Tower defense puzzles evolved from the base defense portion of real-time strategy games which, in turn, were simplifications of real world military battles. Tower defense puzzles are more focused and less complex than the domains they came from. While the following properties are true for most tower defense games, the details are specific to the one used in this research, GopherTD:

- **Perfect Information**. All information is known ahead of time. There is no hidden information during game play.
- **Deterministic**. Barring errors introduced by implementation, GopherTD is deterministic. Creeps move at a constant speed, towers have a consistent way of selecting targets and the shots fired by the towers do a constant amount of damage.
- **Infinite Time**. The problem solver's role is to place towers before the creeps arrive. In GopherTD, as in many other games, the creeps do not show up until the problem solver indicates they are done. The problem solver may take as long as they want to configure their solution.
- **Partially Discrete Space**. The problem solver's only action is to place towers on the map and towers are placed on an 18x22 grid. Creeps, however, move in continuous space and the tower's circular range area is in continuous space. In some tower defense games, the player can place towers in continuous space, but grid-based games are far more common.
- **Non-Adversarial**. Although the goal is to stop the creeps, the creeps make no choices. They follow a pre-determined path and make no attempt to speed up, slow down or dodge, nor can they affect the player or the towers. In this sense, the creeps are no different than parts moving down an assembly line.

3.2.3 Terminology

Strictly speaking, tower defense games are not games. In the terminology of ludology (the study of games), a game must have a goal and an opponent. If there is no end goal (e.g., The Sims, Animal Crossing, Harvest Moon, Farmville), it is a toy. If there is no opponent (e.g., FreeCell, Rubrics Cube, Angry Birds, Lemmings), it is a puzzle. Tower defense games are puzzles. Like most puzzles, a solution that works once is guaranteed to always work. In tower defense, as with some but not all puzzles, there are multiple solutions.

Despite this, we need to be able to differentiate between the class of problems (tower defense), the different versions of tower defense that have been created (Plants vs. Zombies, Bloons, GemCraft, Ghost Hacker, etc.) and the specific problems (i.e. the maps) in each version. We will therefore use the following terms:

- **Tower Defense**. The class of problems.
- **Tower Defense Game**. A specific instance of tower defense. A tower defense game defines the rules, towers, creeps and available maps for that instantiation. This is not a game as defined by ludology but as defined by retailers – it is the product that is packaged together and placed in the video game aisle. The specific game used in this research is GopherTD.
- **Tower Defense Puzzle**. A specific map in a tower defense game. Each map is a unique problem that requires a unique solution. GopherTD has 16 maps, also referred to as puzzles and problems.

3.2.4 Video Games as Experimental Laboratory

While there has been considerable debate about the value of video games (Are they a serious topic for research? Are they of value in education? Are they harmful to society?), there is no question that they are an ideal tool for behavioral research. Games have many benefits as a research apparatus including:

- **Detailed, Automated Data Collection**. Every action one takes in a video game can be recorded. This includes lengthy streams of actions, per-event timing (including idle time), options evaluated and undone (in our case, placing a tower and then deciding to remove it) and even mouse movements (including speed and motion profile of movements).
- **Logging and Replay**. It is possible with a game to record the subject's behavior and replay it for analysis. Actions and outcomes can also be shown to subjects, either in the form the subject experienced them in or placed side-by-side with other recordings, allowing the experimenter to ask the subject detailed questions about actions without requiring the subject to remember their exact actions.
- **Motivation**. Video games can be more motivating than other types of experiments, although many research projects are likely to use simplified games focused on a single topic and such games might lack the features necessary to motivate subjects.
- **Impossible Scenarios**. There are numerous experiments that require more resources than are realistically available to a researcher. Other experiments are dangerous. Examples include many real world tasks such as fire fighting, flying a plane or conducting surgery. These can be (and often already have been) simulated in a video game. These simulations lack the fidelity of real-world situations (haptic feedback, environmental conditions such as heat, etc.), and certain phenomena are known to not translate well to games (e.g., stress levels and reactions to suppressive fire in military simulations) but even low fidelity simulations allow experimenters to collect meaningful data for situations that are otherwise too costly or dangerous.
- **Strict Variable Control**. The experimenter has control of all variables in the experiment, allowing them to include, remove, abstract or otherwise modify variables of interest to the research.
- **Ecological Validity**. Many human studies have been criticized for using unrealistic (i.e., ecologically invalid) tasks, leading experimenters to find

phenomena that don't exist when real world (i.e., ecologically valid) tasks are used (see, for example, Kahneman and Tversky 1996; Gigerenzer 1996). Games, at least commercial ones, must fit human capabilities in order to be successful. Too easy and the player becomes bored, too difficult and the player becomes frustrated. Good games align with the abilities of the player.

3.2.5 Where Tower Defense Fits in the Problem Space

In the last few years there have been many papers published on artificial intelligence in real-time strategy games. These games are typically simulations of military battles. The players gather resources (wood, metal, etc.), use the resources to build factories, use the factories to build military units and use the military units to destroy the other players' headquarters. They must do this while trying to protect their own headquarters. Tower defense grew out of this last part.

The first tower defense games were modifications to existing real-time strategy games such as Warcraft and Starcraft. The player was responsible for organizing the defenses around their base, which involved placing walls and guard towers along the path to the base. These restricted games became popular enough to spawn their own category, tower defense. Since then, hundreds of tower defense games have been created. This is in large part because of hobbyists; tower defense games are simple enough that a single person can make one in a reasonable amount of time with no money, which is not true of real-time strategy games or most other popular game types. The majority of modern TD games are free-to-play, Flash-based Web games.

We believe GopherTD to be an optimal level of simplicity. The low level of complexity makes it easier for subjects to learn the task and for us to analyze the data but is still complex enough to allow meaningful study of several key ideas of value to more complex domains including spatial representation, spatio-temporal reasoning, strategy use and transfer learning.

3.2.6 Match to Criteria

In 3.1 Desired Properties of a Domain we defined the criteria we used to pick a domain for this research. Tower defense is an excellent match for these criteria, as shown below.

3.2.6.a Goal Criteria

Predominantly Spatial Actions. There are only two types of actions in tower defense: selecting which towers to use and where to place them. In our third experiment, we gave subjects the towers and the only decision subjects could make is where to place the towers on the map.

Non-Spatial Goals. The goal of a tower defense problem is to prevent creeps from reaching a location. The score is equal to the number of creeps that are stopped.

Meaningful Space-Time Interaction. One of our original criteria was to find a domain that involved objects moving through space over time, which is obviously true of tower defense. How significant the space-time interaction proved to be was not apparent until later in the study. A detailed explanation of this is given in Chapter 6 Unifying Spatial and Temporal Reasoning.

3.2.6.b Realism Criteria

Complexity. Tower defense is no less demanding than real-world tasks for spatio-temporal reasoning and offers just as much opportunity for strategic reasoning and transfer learning. Below we discuss how tower defense is an extreme simplification of real-world tasks but these simplifications – deterministic actions, discrete space, single-task focus, etc. – do not affect the complexity of the issues we wish to investigate.

Definition. We wanted a domain where, as in most real-world situations, the important features were hard to define. Unlike most real-world environments, the raw data in tower defense is precisely defined – space for tower placement is discrete, the speed of creeps is uniform, the damage done by towers is fixed, etc. The only variables the problem solver can directly influence are the placement location of the tower (on an 18x22 grid). These features are precisely defined and represent what the problem solver can control but they are not the important features. Success in tower defense relies on a large set of derived properties and meta-properties, all of which the problem solver must define for themselves. Chapter 7 Solution Representation lists 13 types of representational concepts covering more than 100 representational features observed in our experiments. Chapter 9 Computer Model covers several additional representational features that needed to be created to implement subject strategies. Identifying these additional features is critical to problem solving in this domain.

Variation. In the real world, people frequently encounter problems that are different enough from previously solved problems that they cannot re-use or adapt known solutions but similar enough that they can use something from their experience to solve them. One of the goals of this research is to determine what that something is and how this ability to leverage experience works. Tower defense is an excellent domain for this. The problems in tower defense are maps and these maps are dissimilar enough that we see no way to re-use known solutions or models. Despite this, our research has shown that, after training, humans are able to solve novel tower defense problems as quickly and accurately as ones they have previously solved, indicating strong transfer performance. This is one of the more appealing aspects of this domain.

3.2.6.c Performance Criteria

Ecological Validity. We wanted a task that was a good match for human capabilities. This is a strong argument for why popular video games make excellent research platforms. For a commercial game to become popular, it must be at the sweet spot of human ability. If a game is too difficult, the game is rejected for being frustrating, and if it is too easy, the game is rejected for being boring. In this work we use a research version of Vector TD, a game that has been played several million times (Flash Element

TD, the first version of the game containing a single map, was downloaded 100 million times in its first year). At the time this research began, Vector TD was the 8th most popular tower defense game (out of 252) at the Web site towerdefense.net

"Average Human" Success. We categorized subjects into two groups, domain novices (people who knew nothing about tower defense) and domain experienced (experience playing tower defense games). Most domain novices did not play video games of any kind. While subjects with domain experience had a higher average score than domain novices, domain novices significantly outperformed a baseline software agent we developed (details are given in Chapter 11 Results of Experiment 2). On easier problems, almost all domain novices earned perfect scores. We believe these results show that humans, even those with no experience in this domain, are good at this task.

Humans Significantly Outperform Computers. This criterion is hard to verify as the work presented here is the first to use this domain. Much work has been done in a related domain, real-time strategy games (see, for example, (Buro 2003; Weber, Mateas, and Jhala 2011)), and humans still lead computers in that domain but as it is a more complex domain it offers little evidence about the tractability of tower defense. We therefore offer these observations:

- **No Meaningful Action Sequence**. Unlike games such as chess and Pac-Man, there is no sequence of moves in tower defense and no intrinsic notion of an intermediate state. This makes it difficult to use techniques such as minimax, Markov Chain Monte Carlo and Upper Confidence-bound on Trees, which examine a sequence of actions.
- **Slow State Evaluation**. In some domains, the search space is so large that even with current supercomputers the problems would take decades to solve. This is not the problem with tower defense. The time it takes to search a state space is related to both the number of nodes to search and the time it takes to process a node. In problems such as chess, constraint satisfaction and classic puzzles such

as missionaries and cannibals, it is trivial to determine whether a given state is the goal state. This is not true for tower defense. To determine whether a given configuration of towers will achieve a perfect score, a simulation must be run. In GopherTD, we evaluate states by running the simulation at 50 times normal speed (in our implementation, running faster than that introduces slight timing errors; given the temporal nature of this problem, slight variances in time often have large effects on outcomes). The process takes a minimum of four seconds. Even in small state spaces, evaluating all states is impractical. There are currently no heuristics known for this domain that would reduce the size of the search space.

Human Data is Meaningful to Computers. Before we began this work we had some ideas on how humans succeeded in this domain and believed it relied on mechanisms that could be implemented usefully in computers. We have since learned quite a deal about how humans work in this area and while many of our initial intuitions were simplistic, we have shown in this work that the mechanisms used by humans is of value to computers.

3.2.6.d Technical Criteria

Simple. Tower defense is a simplification of real-time strategy games which are, in turn, simplifications of military battles. Tower defense is less complex than real-time strategies in two ways.

First, tower defense lacks many of the properties that make problems hard (for a discussion, see Section 3.2.2 Properties). Actions are deterministic, states are discrete and no information is hidden or unknown. The size of the maps is considerably smaller than real-world problems (GopherTD is played on an 18x22 grid, considerably smaller than real-time strategy maps, where even small maps are 4,096x4,096). While some genres of games have time pressures and require a moderate amount of manual dexterity, tower defense is among the many types of games that have no time pressure and make no demands on hand-eye coordination.

Second, in contrast to real-time strategy games, tower defense focuses on a single task, defending the base. The ability to solve multiple tasks concurrently is an important and interesting ability and a relevant future extension to this work. Given that the time, attention and working memory pressure in real-time strategies has a presumably important impact on decision making, we wondered if tower defense was too simple. Our findings show we were right in choosing this domain: the domain is complex enough to support our study of how humans solve spatio-temporal problems.

Focused. Many games require the player to perform multiple, dissimilar tasks. For example, in real time strategy games, from which tower defense games were derived, the player must solve problems related to combat, building, resource management and technical research. Tower defense is one of only a few game types that have a single task. Other single-task game types include simpler arcade games (e.g., Pac-Man), platforming games (e.g., Super Mario), fighting games (e.g., Soul Caliber) and vehicle games (e.g., Microsoft Flight Simulator). Of these, tower defense is the only one that does not rely on hand-eye coordination and also is not performed under time pressure.

Configurable. One of the advantages of using a video game or other computer task as a research tool is that it is often easy to configure the experimental apparatus to test a specific hypothesis. Tower defense is particularly easy to configure as tower defense problems are simply maps and configuring maps merely involves placing the walls, paths and entry and exit points on a small grid. The tower defense domain can be quickly configured to test a variety of theories.

Fast. We conducted three sets of experiments. Each experiment involved a training phase, testing phase and after-action review. Subjects solved 12-16 problems in the testing phase and 20-48 problems in training. Running one subject through an experiment took 2-3 hours. The tower defense task was quick enough that we were able to collect detailed data on a sufficiently broad set of problems in a reasonable amount of time.

Motivation. Our experiments used a research version of the tower defense game Vector TD. Vector TD is an extremely popular commercial video game and is considered to be intrinsically motivating. Motivating subjects to perform their best did not appear to be a problem in our experiments.

Previously Unknown. Most of the subjects in phases 1 and 2 were familiar with the tower defense domain. They were not, however, familiar with the specific tower defense game we used. As these puzzles were new to the subjects, they were required to reason out a solution to each puzzle rather than recall a solution from a previous experience.

Minimal Leveraging of Real-World Knowledge. Tower defense came from the base defense portion of real-time strategy games, which came from real-world military battles. This makes it possible for subjects to leverage real-world information, which we did not want in these experiments. The specific tower defense game we used was far more abstract than most tower defense games. Enemies were geometric shapes rather than tanks, soldiers, aliens, etc. Towers were colors rather than rocket launchers, machine guns, etc. This appears to have minimized the subjects' use of external information, although there were some cases of inappropriate, negative transfer. Specifically, a small number of subjects focused on bottlenecks, a spatial feature which is important in real-world military battles but of no value in our task. Finding a task that has ecological validity but does not allow leveraging inappropriate external knowledge is not easy but we believe that GopherTD did as well as can be expected.

Multiple, Qualitatively-Different Instances. These puzzles satisfy the criteria we were interested in, namely that the new problems were sufficiently different that previous solutions could not be used or adapted to solve the new problems but were similar enough that the problem solver could solve the new problem better (i.e., more quickly or accurately) because of their previous experience. The ability to leverage experience to solve novel problems was an important part of this research.

3.3 Task and Tool: GopherTD

3.3.1 GopherTD: Task

This section describes GopherTD, the tower defense game used in this research, at the same level of detail that was given to subjects.

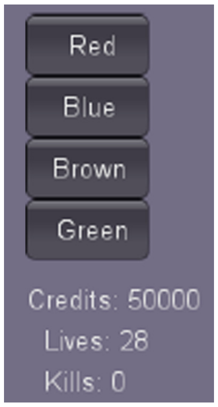
Tower defense has strong spatial and temporal components, making it difficult to understand from just text. A step-by-step example is shown over the next two pages.

Task – Phases, Goal, Scoring. In a tower defense game, the problem solver is given a map containing an entrance, an exit and a set of walls forming a simple maze. A set of inanimate objects known as creeps follows a pre-determined path from the map entrance to the map exit. The path runs through the maze of walls. The problem solver can place towers on the walls. The towers damage creeps that move into their range. When a creep takes too much damage, it is destroyed and removed from the map.

The goal of GopherTD is to prevent the creeps from reaching the map exit. There are two phases. During the build phase, the problem solver purchases a set of towers and places them on the walls. During the execution phase, the creeps enter the map and move along the path and the towers attack the creeps that come into range. The problem solver cannot take any actions during the execution phase. If the problem solver has placed their towers at good locations, all of the creeps will be stopped.

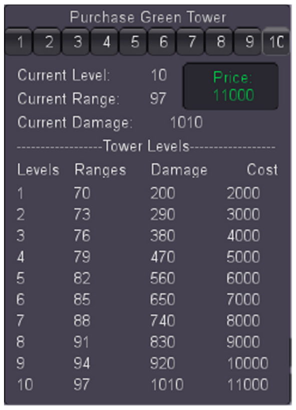
The score is the number of creeps that are prevented from reaching the exit.

Planning Phase



Credits: 50000
Lives: 28
Kills: 0

You have $50,000 to purchase towers.
We start by selecting the Green (fast attack) tower.

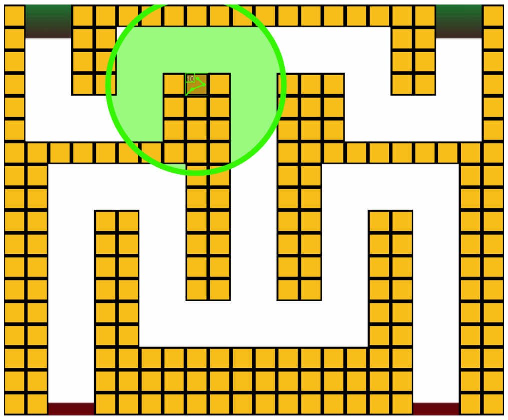


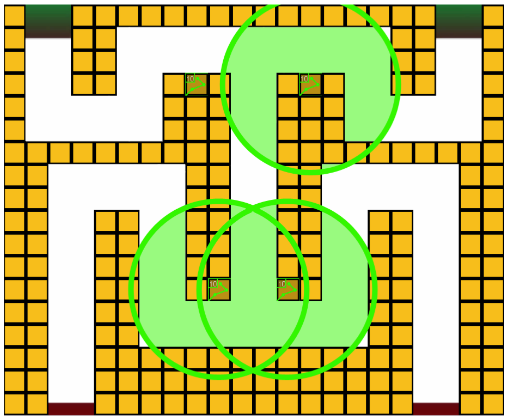
Current Level: 10
Current Range: 97
Current Damage: 1010
Price 11000

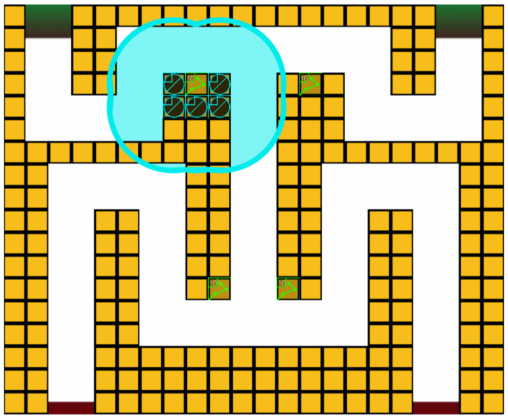
-----------------Tower Levels-----------------

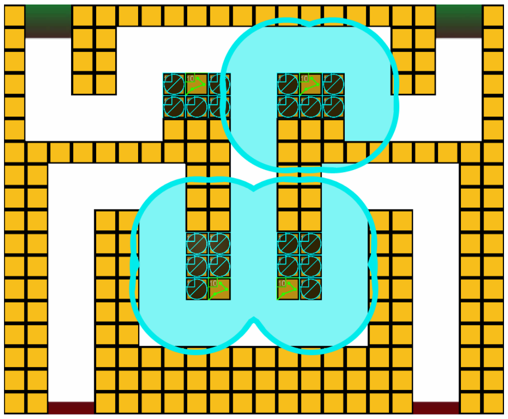
Levels	Ranges	Damage	Cost
1	70	200	2000
2	73	290	3000
3	76	380	4000
4	79	470	5000
5	82	560	6000
6	85	650	7000
7	88	740	8000
8	91	830	9000
9	94	920	10000
10	97	1010	11000

We choose the upgrade level of the tower.
We select the highest level, level 10 ($11,000).


We place the tower at the end of a U-turn so that the tower can cover a lot of area.

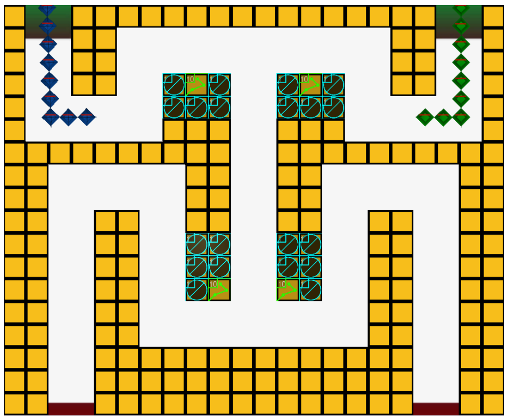

Buy 3 more level 10 Green towers. Place them on other U-turns.

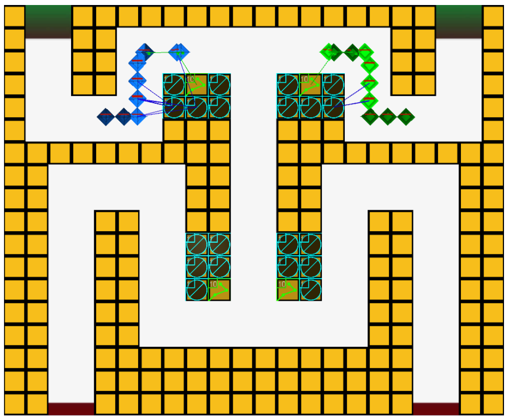

Buy 5 level 1 Blue (slowing) towers ($300 each). Place them around the first attack tower.

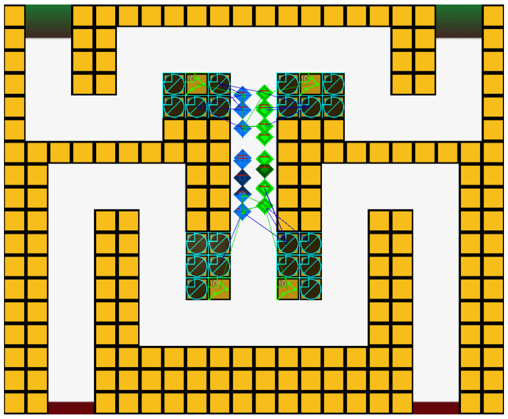

Place Blue slowing towers around the other towers.

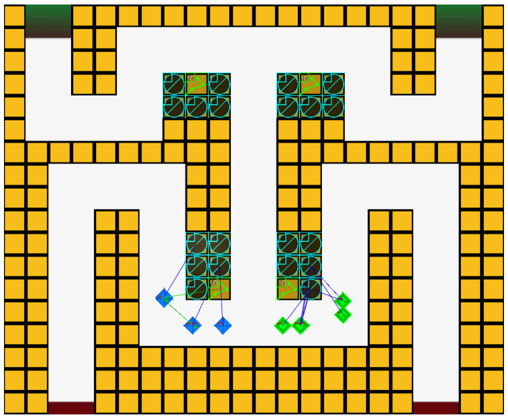
All the money has been spent. The planning phase is now over and the execution phase begins. The problem solver cannot take action during the execution phase.

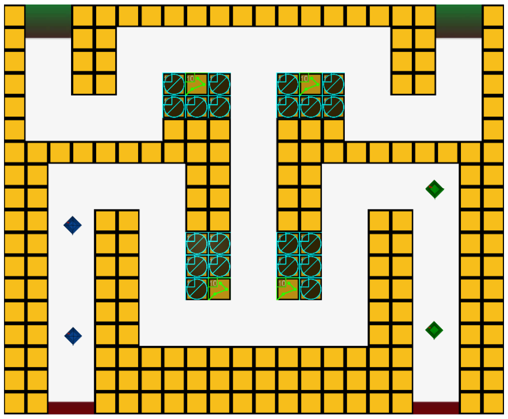
Execution Phase

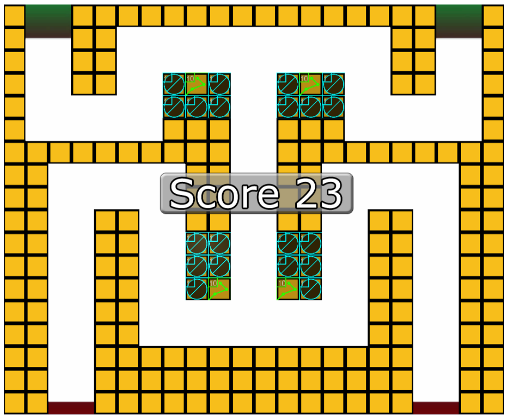

The creeps enter from the starting positions.


The creeps enter the range of the first towers. Lighter-colored creeps move at half speed for 2 seconds.


Creeps move down the center. 8 creeps have been destroyed so far.


Creeps move past the second set of towers. 18 creeps have been destroyed.


Creeps are out of range and can no longer be stopped.



Final score: 23 out of 28 creeps were stopped.

Creeps. There are 28 creeps, two lines of 14 creeps each. Each line of creeps was assigned a different color (green and blue) to help subjects track each line. The color has no other impact. All creeps are identical in speed and health.

Towers. There are 11 types of towers. Their names are based on their type (represented by colors; powerful red, fast green, slowing blue and mixed purple) and relative power (one through three with the exception of blue, which has two options). Towers differ in cost, number of creeps they can affect at one time, the number of shots they fire per second, range, ability to damage creeps and special powers. Power and range can be upgraded for an additional cost. Upgrading one level costs half the base tower cost (e.g., a $100 Green1 tower costs $50 per upgrade) and increases damage by a little less than half the base tower's amount and range a small, fixed amount. There are 10 upgrade levels for each tower and therefore 110 types of towers. The strongest tower, the level 10 Red3, can destroy one creep in three shots.

Towers must be placed on a wall tile. Only one tower may be placed on a given tile.

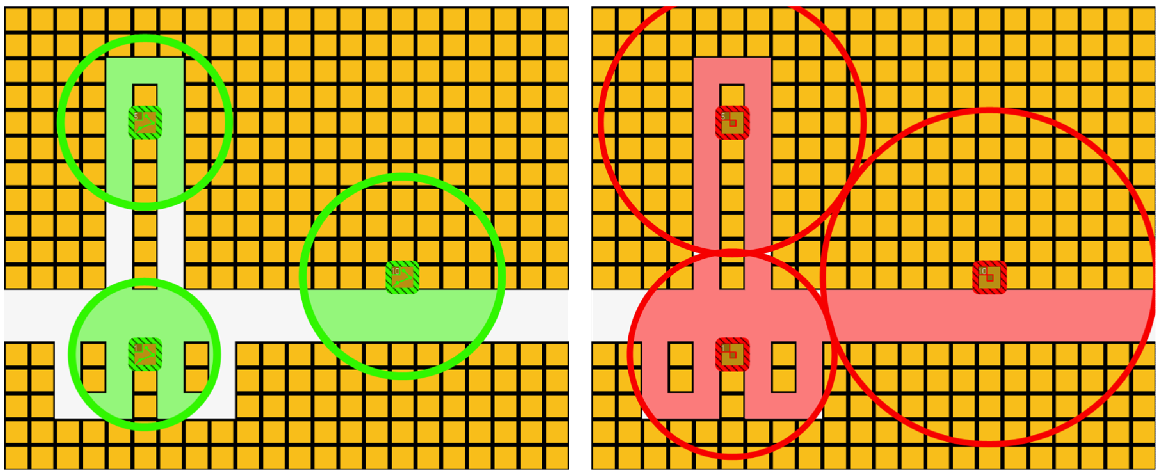


Figure 3.1. Tower Ranges. LEFT: Ranges for Green1 towers level 1, 5 and 10. RIGHT: Ranges for Red3 towers level 1, 5 and 10.

While there are many ways to think of towers (several of which are listed in Chapter 7 Solution Representation), the most important distinction is between offense towers and slowing towers. There are nine types of offense towers (3 green, 3 purple and 3 red) and two types of slowing towers (blue and brown). When subjects talk about their strategies, they often refer to offense and slowing towers rather than the more specific tower types. We follow the same convention in this document.

Purchasing Towers. Because certain strategies can only be carried out with certain towers, when investigating the number and type of strategies used, we allowed subjects to select their own towers. Subjects were given $50,000 and allowed to select any number of towers from each of the eleven tower types at any of the 10 upgrade levels. Tower prices ranged from 100 (level 1 Green1) to 15,400 (level 10 Purple3). The number of towers a subject could have ranged from three (for level 10 Purple3) to 500 (for level 1 Green1).

Targeting. When deciding which creep to attack, the tower chooses the creep closest to it (assuming the creep is in range). Towers in GopherTD use sticky targeting – once a tower has selected a target, the tower only fires on that target until the target is either destroyed or out of range.

Maps. There are 16 maps in GopherTD. All maps are 18x22 grids. Maps have either one or two entrances and exits. The length of the path through a map differs from map to map, from 50 tiles to 244.

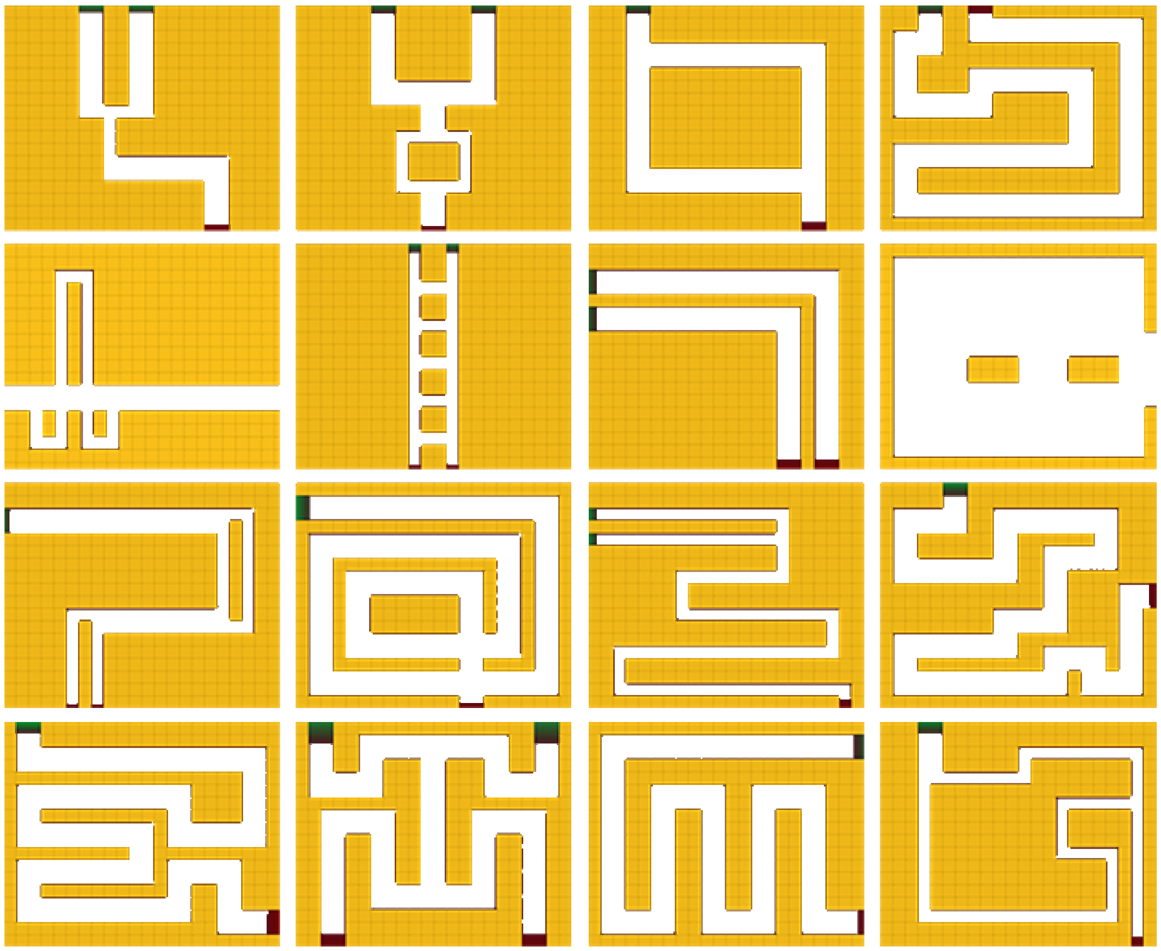


Figure 3.2. The 16 Maps in GopherTD.

Chapter 4

Experiment 1

Human Problem Solving

Before we could build computer agents with the human-like capacity to solve novel, tower defense puzzles, we needed to understand how humans did it. Our primary interest in Experiment 1 was to get a feel for how humans solved these kinds of problems, with more detailed investigations carried out in later experiments.

The scope of Experiment 1 included capturing the variety of representations and problem solving mechanisms, which is the focus of the research presented here, but the initial focus of the GopherTD project included how humans learned from experience and how they leveraged past problem solving episodes to more rapidly solve novel problems. This focus is evident in the experiment protocol, where subjects were asked to solve a number of novel problems and given multiple attempts to solve two of the harder problems. We conducted this study with the belief that the three topics – how problems are solved, how learning occurs, and how knowledge is transferred – were rooted in the same fundamental mechanisms.

Our research is, to the best of our knowledge, the first to investigate this domain and the first to study complex spatio-temporal problem solving at a detailed level. As a result, it was unclear where to start and what to measure. Experiment 1 cast a wide net, with different maps used to study problem solving, transfer learning and learning from experience. We therefore decided to conduct the study in two phases. The goal of Phase 1 (n = 13) was to collect initial data that could be used to create more focused experiments. These lessons were used to create more effective tests for Phase 2 (n = 25).

4.1 Phase 1

Category	**Count**	**Male**	**Female**
Experienced	12	11	1
Novice	1	1	0
Total	13	12	1

Table 4.1. Demographics, Experiment 1 Phase 1.

Subjects first trained on a set of two training tower defense puzzles and then solved a set of eight tower defense puzzles, which included the two training maps, under a think aloud protocol. Upon completion of the task, subjects participated in an after-action review (see Section C.1.3 Think Aloud Protocol for more information).

Training. The two maps shown in Figure 4.1 were used for training. Based on difficulty rankings assigned by Vector TD, both training problems were categorized as medium difficulty. Subjects could practice solving each training problem as many times as they wanted and were free to switch between problems as often as they liked. In order to study subject strategies, we wanted subjects to train until response stabilization. Subjects were asked to practice until they felt comfortable solving these types of problems. The average subject spent 86 minutes creating solutions for 17 problems in training.

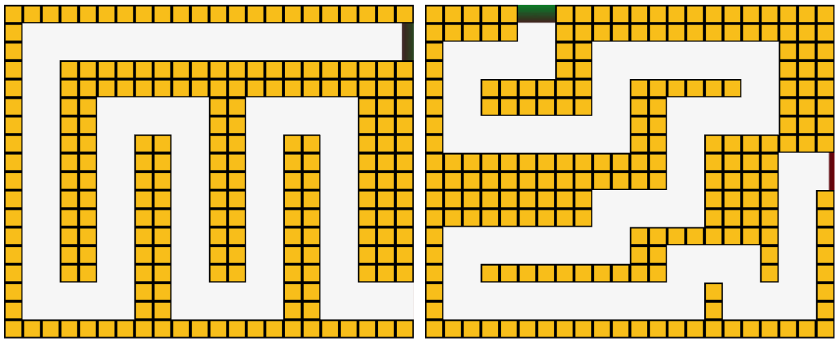


Figure 4.1. Training Maps, Experiment 1 Phase 1.

Training	Min	Mean	Max
Trials	7	16.67	38
Minutes	60	85.55	122

Table 4.2. Time Spent Training, Experiment 1 Phase 1.

Testing. Of the eight maps used for testing the first two maps were the training maps, used to establish a baseline. Maps 6 and 8 were considered high difficulty. Map 7 (Void) was either medium or high difficulty, depending on the strategies used by the subject. Because of the significant difference in the length of the two paths, many subjects learned STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING either upon attempting or viewing map 7. Subjects who used the strategy for the first time on map 7 were highly likely to use it on map 8. While we appreciated the spontaneous learning and transfer, it worked counter to our goal of response stabilization.

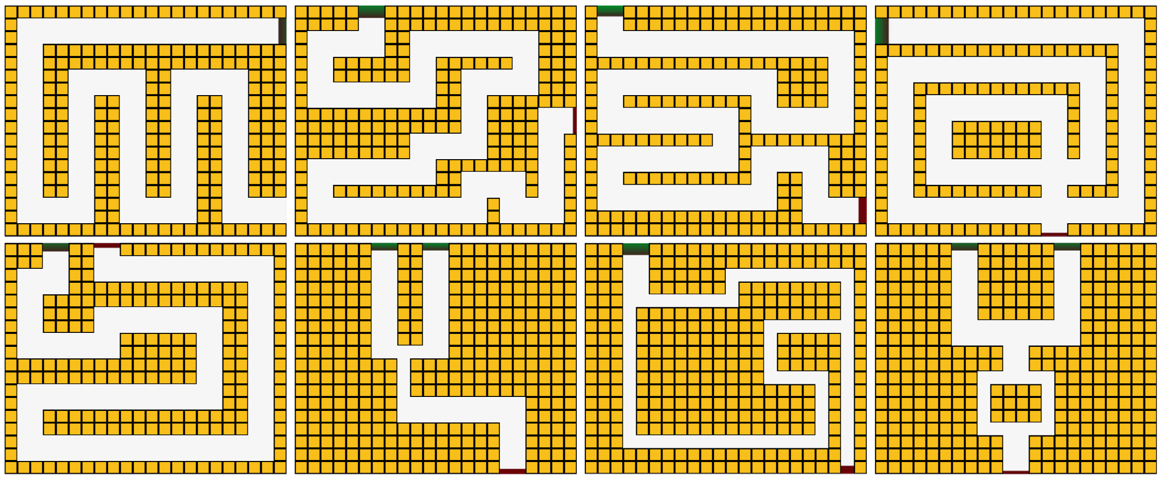


Figure 4.2. Testing Maps, Experiment 1 Phase 1. The first two (top left) were both training and test maps and used to study problem solving. The next four maps were new and used to study transfer learning. The final two maps (bottom right) were used to study learning.

Subjects were given one chance to solve problems 1-6. They were given up to five attempts on problems 7 and 8. As in all experiments in this work, the maximum score was 28, representing the number of creeps stopped.

No break was given between training and testing.

In Phase 1, the trial ended when five creeps made it to the exit. While consistent with how all tower defense games work, this end-point criterion had the potential to lower scores; of 28 creeps, if the first five made it to the exit, the subject still had a chance to stop one or more of the remaining 23. The end-point crtierion was changed in later experiments, with trialsstopped only when all 28 creeps had escaped or been destroyed.

Problem Solving. Scores on the first two maps, which were used also in training, represented the subject's ability to solve GopherTD puzzles given unlimited time. We were less concerned with the score they earned than the process they used. We used this information to build the catalog of strategies and representations presented in Chapters 5 and 6.

Transfer. Maps 3-6 were not previously known to the subjects. They were given a single chance to solve each. Problems were sufficiently different that subjects could not re-use a solution, even in a qualitative way. They had to solve each problem using only the useful knowledge gained from solving the training problems.

Problems 3 and 4 were easier than the training problems. Problem 5 was of the same difficulty. Problem 6 was much harder.

Learning. As with maps 3-6, subjects had not previously seen maps 7-8. Map 8 is difficult, of equal difficulty as map 6 for similar reasons. The difficulty of map 7 depended on the strategies the subject used. For those who knew STRATEGY TP8: EXPLOIT GEOMETRY, this map was of moderate difficulty. For those who did not, the map was considered quite hard.

After attempting each problem, the subject was allowed to try four more times. We used this opportunity to study how subjects adjusted their approach to a problem given feedback. In practice, after each trial the subject evaluated whether they could do better and stopped once they felt they had done as well as they could. Given the difficulty of the problems, most subjects made several attempts at each problem, allowing us to gather data on learning and optimizing solutions.

4.2 Phase 2

Phase 2 was a refinement of Phase 1. In Phase 1, 13 subjects had solved the problems in 13 different ways, displaying far more variety than we originally believed the domain would support. Recognizing that we had not explored the full solution space of this domain, Phase 2 focused on gathering more data on the different ways the problems could be viewed, reasoned about and solved.

Several changes were made for Phase 2. As we were no longer focused on how humans learned from experience, we no longer allowed subjects multiple attempts to solve training maps. The map Void was made a training map. In Phase 1, this map caused subjects to learn new strategies that they applied to later problems. This lack of consistency made it difficult to analyze each subject's problem solving approach.

While transfer performance was no longer the focus of the work, we continued to use separate training and test problems as we believed that the ability to leverage experience to solve novel problems was important to understanding all problem solving in this domain. We also noted that subject scores on novel maps were similar to their scores on training maps of the same difficulty, implying that studying performance on novel problems did not interfere with the goal of collecting data on strategies and representations.

Category	Count	Male	Female
Experienced	17	16	1
Novice	2	0	2
Author	2	2	0
Special	4	1	3
Total	25	19	6

Table 4.3. Experiment 2 Demographics.

25 subjects participated in Phase 2. 17 of these had experience with other tower defense games but not GopherTD. In Phase 1, one novice subject was tested, requiring 8 hours to finish the experiment. We tested two additional novices in Phase 2 to determine whether the original case was unusual (it was). Data was also collected on the authors of this experiment and four lab members who had heard talks about this research. Since the goal was to capture the diversity of approaches available in the domain rather than comparing performance, we deemed these subjects appropriate to this experiment.

In Phase 1, 8 maps were used but two were played 10 times for a total of 16 trials plus training. In removing the learning portion of the experiment we removed more than half the testing trials. This allowed us enough time to use additional maps in Phase 2. In this experiment, five maps were used for training and 12 maps for testing.

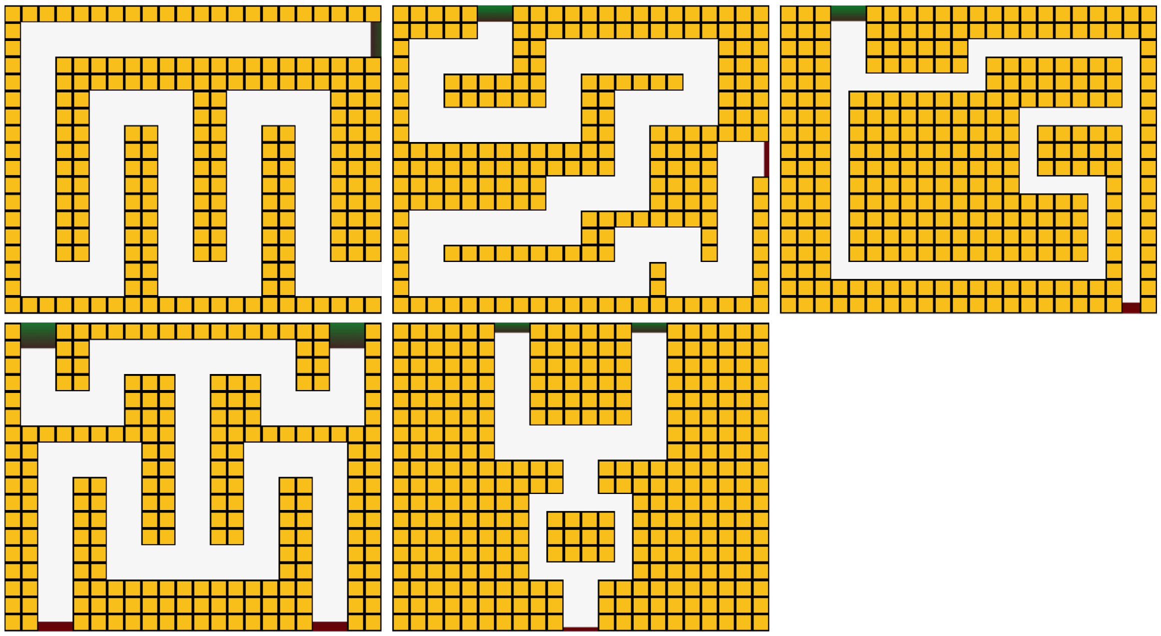


Figure 4.3. Training Maps, Experiment 1 Phase 2. Maps 1, 2 and 4 are medium difficulty. Map 5 is hard. The difficulty level of Map 3 (Void)depends significantly on the strategies the subject has learned.

Training. Subjects trained on five maps: three of normal difficulty, one of high difficulty and Void. As mentioned in Phase 1, the difficulty of Void depended on whether the subject knew one of the temporal strategies that exploited path synchronization gaps (i.e., STRATEGY TP8: EXPLOIT GEOMETRY or one of the DIFFERENTIAL SLOWING strategies; these are discussed in Chapter 5 Strategies). Only one subject used these strategies in Phase 1 prior to seeing Void. After seeing Void, six subjects (46%) used a DIFFERENTIAL SLOWING strategy. Given how important this map was in forming subject's strategies, it was made a training map.

Training	**Min**	**Mean**	**Max**
Trials	7	15.16	39
Minutes	12	74.87	150

Table 4.4. Time Spent Training, Experiment 1 Phase 2.

As before, subjects could practice solving each training problem as many times as they wanted and were free to switch between problems as often as they liked. Subjects were asked to practice until they felt comfortable solving these types of problems. Despite the larger number of training maps, the average time spent in training decreased slightly from 17 problems over 86 minutes to 15 problems over 75 minutes.

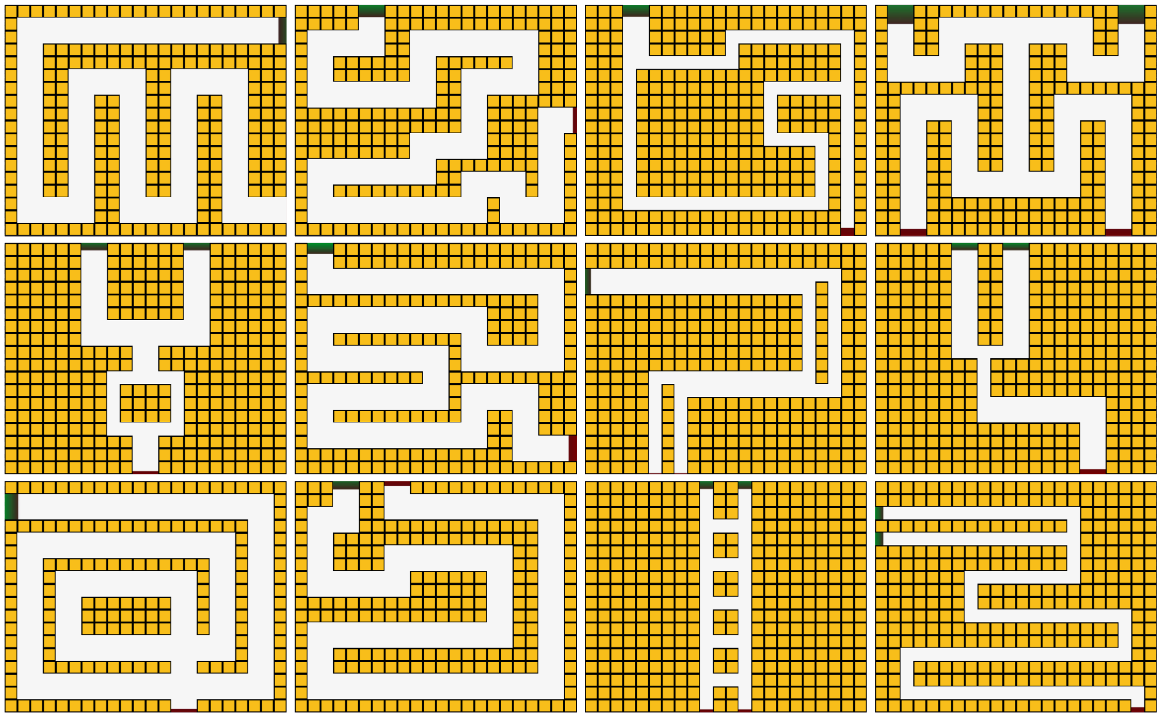


Figure 4.4. Testing Maps, Experiment 1 Phase 2.

Testing. 12 maps were used for testing. The first five maps were the same as those used in training. Subjects were given a single chance to solve each puzzle. The maximum score was 28 and represented the number of creeps stopped. The trial ended when all 28 creeps either escaped or were destroyed, a procedural change from Phase 1.

Of the five training maps, one was of low difficulty, three were of moderate difficulty, one hard and one variable. Of the remaining seven maps, maps 9 and 12 were of low difficulty, maps 6 and 11 were of moderate difficulty and maps 7 and 8 were of high difficulty for a total of three low difficulty, five moderate difficulty and three high difficulty. In order to maintain subject confidence and motivation, we placed the difficult maps in the middle of testing, followed a hard map (8) with an easy one (9) and ended on an easy map (12).

No break was given between training and testing.

Chapter 5

Results of Experiment 1

The goal of Experiment 1 was to determine how humans solved complex spatio-temporal problems. From observations gathered in Experiment 1 we learned several things.

Affordances, not Features. The subject's goal is not to describe space, it is to solve a problem using it. Asked to describe a problem, subjects didn't talk about the spatial features of the map, they talked about the exploitable properties. We follow the standard psychological practice of referring to these as the map's spatio-temporal *affordances*. Recognition of affordances depended on the structure of the map, the capabilities of the towers the subject used and the strategies known by the subjects. Affordances are discussed in Section 5.5 Spatial Representation.

Strategies and Strategy Sets. Subjects solved problems using sets of simple, focused, affordance-specific strategies. Types of strategies included tower selection strategies, placement strategies, learning (exploration) strategies and strategy selection strategies. The work discussed in this paper focused on placement strategies. We identified 78 strategies related to where towers are placed and divided them into the categories: spatial, temporal, satisficing, targeting, efficiency, polish and other. The full list of placement strategies is given in Chapter 8 Strategies.

Focused, Integrated Reasoning. Strategies exploited the affordances subjects identified in the problem. Affordances often were created by noticing events during execution. For example, a subject might notice that towers that cover more path area get higher scores and that towers on U-turns cover more area than towers placed in other areas. U-turns become an affordance, a property of a map that can be exploited. When presented with a new problem, the subject immediately looks for U-turns and, if they're found, ignores every other property of the map. The subject then uses the strategy "place towers on U-turns" to place the towers. These pieces are interrelated – looked at in isolation, it is difficult to understand why a U-turn is an affordance or why a trivial strategy such as "place towers on U-turns" works. The elements must be examined at the system level, each element in the context of the others. For this reason, we begin this chapter with an integrative look at one subject's solution, given in Section 5.1 Solution Analyzed: The Problem Solving Approach of E1500.

Representational Elements. To describe subject solutions, we divided information into five components: tower selection, role, functional group, placement decision and distribution. These components have several subsections. A particularly rich source of representational elements was placement anchor, which is the property that something is placed relative to (e.g., U-turn, beginning of path, center of map, etc.). We identified 37 affordances in nine categories. An overview of the types of representational elements discovered is discussed in Section 5.3 Components of a Solution Attempt. The full list of representational elements is given in Chapter 7 Solution Representation.

Representation as Reasoning. One of our goals was to implement a computer agent capable of human-level performance on complex spatio-temporal problem solving. The observation that had the largest effect on this goal was that most of the intelligence of the human problem solvers in this task was in how they represented the problem. Once the right affordances had been found, creating a strategy to exploit them was normally trivial. Creating a computer system to identify the affordances, however, turned out to be a non-trivial task. For insights into the complexity of defining affordances, see Section 5.5.1 Features, Properties and Relationships vs. Affordances.

Spatial Proxying. How does one solve a tower defense problem? The most common answer was to place towers where they covered the path area. This concept seemed so obvious that it took us a while to realize the mismatch between the goal and action. To maximize the score, a player must maximize the amount of time a tower has to fire, which is a temporal goal, but the strategy of placing a tower where it has the highest usable range maximizes space. Subjects were implicitly and often unconsciously converting a temporal problem into a spatial one and solving that. This led to two observations. First, subjects used space as a proxy for time because it worked remarkably well. Second, it was hard not to use spatial proxying, but those subjects who could avoid such proxying produced effective but often counter-intuitive solutions. The relationship between time and space is discussed in Chapter 6 Unifying Spatial and Temporal Reasoning. The performance of spatial and temporal strategies is the subject of Experiment 2, discussed in Chapter 11 Results of Experiment 2.

5.1 Solution Analyzed: The Problem Solving Approach of E1500

Problem solving involves trying solutions, noticing small but salient events of interest, classifying these events as problems or opportunities, forming one or more representations to describe or explain the event, creating simple goals based on these (i.e., solve the problem), creating simple solutions for the problems and then trying a new solution, repeating until the problem solver is happy with the results. Like the problem solving process itself, it is hard to adequately explain any one of these subprocesses without the context provided by the others. For this reason, we begin by walking through an example.

What follows is a description of how subject E1500 solved the problem (map) Up and Down. As we walk through the example, we will note the observations, beliefs, representations and strategies that the subject uses and show how they are inextricably interrelated.

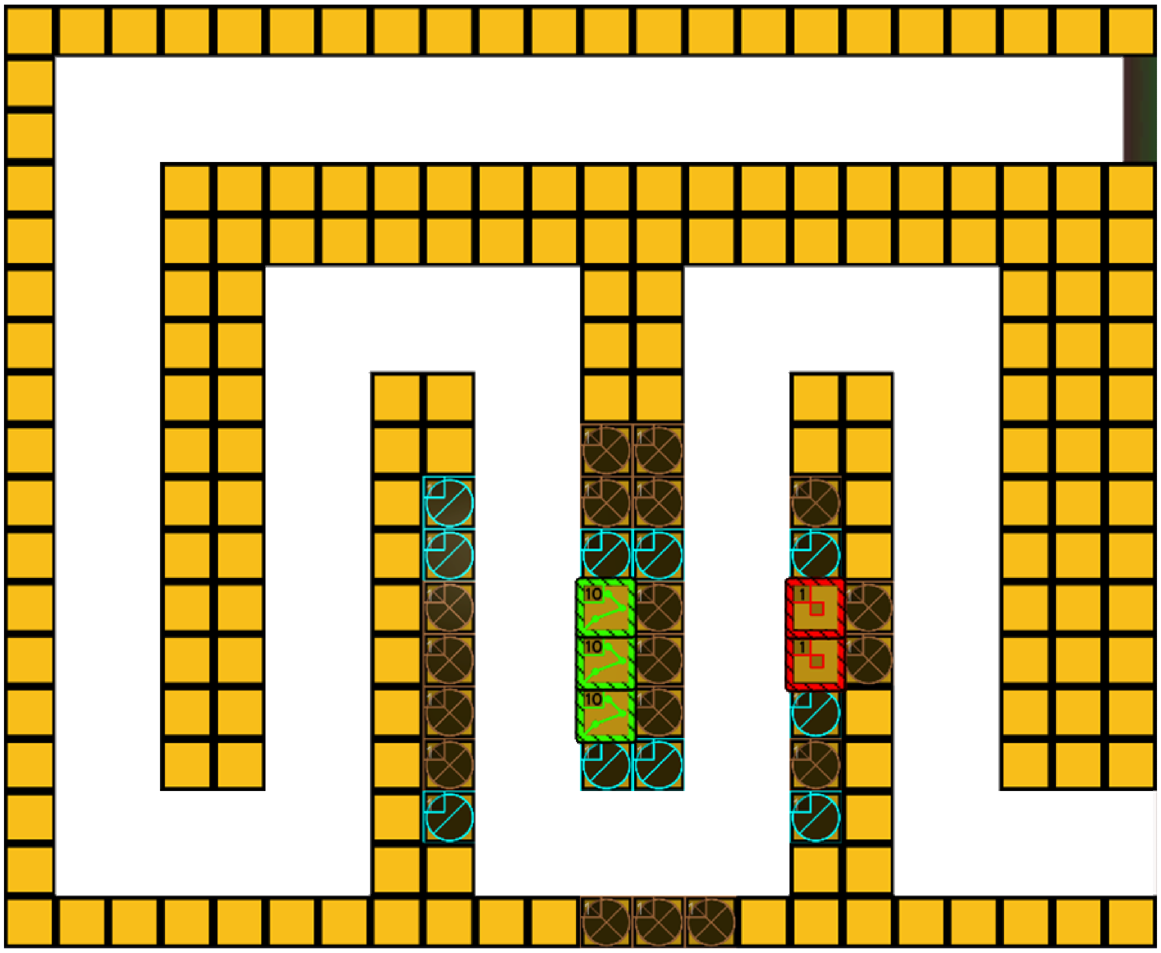


Figure 5.1. Subject E1500's Solution to Up and Down.

The subject used the following towers:

Count	Type	Description
3	Level 10 Green3	These fully upgraded fast attack towers are the core of the subject's offense.
2	Level 1 Red3	These slow-firing but powerful attack towers are used to finish off wounded creeps.
18	Level 1 Brown	These stun towers prevent one creep from moving for two seconds.
10	Level 1 Blue	These slowing towers cause four creeps to move at half speed for two seconds.

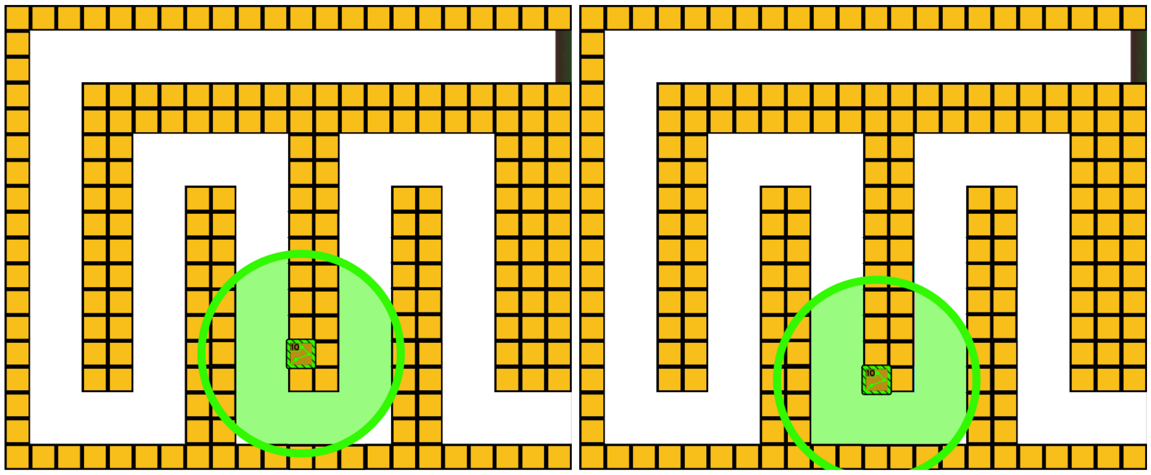


Figure 5.2. Subject E1500's Solution, First Placement, Primary Offense Tower. LEFT: E1500's first tower placement, one space away from the end of a U-turn. RIGHT: An alternative position used by many subjects but not E1500. The end of the range area inefficiently extends into the left and bottom walls.

The first tower placed is a primary (heavy) offense tower. While subjects varied significantly on almost all criteria measured, beginning by placing the favored primary offense tower was common for the majority of subjects.

Why did the subject place the tower where they did? There were several considerations.

Many subjects placed level 10 Green3 towers at the end of a U-turn. Subject E1500 placed the tower one space away from the end of the U-turn. Many subjects noted that towers on U-turns cover more path area than towers on normal wall segments (the towers see three path segments instead of two).

Observation 1	Towers on U-turns cover more area than towers on normal wall positions.
Classification	Opportunity. Can cover more path area.
Strategy 1	Place towers on U-turns to obtain more usable range.

Only a subset of those who noticed the spatial advantage of U-turns also noticed that, on the columns in Figure 5.2, if a level 10 Green3 is placed at the end of a U-turn, the range goes past the path area on two sides, with the end of the range wasted on covering wall area. This information is available to all subjects but only some explicitly recognized it. All of the subjects who remarked on this stated that it was a problem (wasted range). All subjects who remarked that it was a problem moved their towers back one space from the U-turn. This solves the waste problem on one of the two sides. The other side wasn't mentioned, presumably because there is no simple fix other than not using that particular tower (non-upgraded towers have a smaller range) and the subject had already committed to that tower.

Observation 2	Tower's range goes significantly (>= 1 tile) into wall.
Classification	Problem. It unnecessarily wastes usable range.
Strategy 2	Move tower back to avoid wasting usable range.

The central column of the map was picked because the subject knew something of their future plans. They didn't know exactly which towers they'd use or where precisely they'd go but they knew they wanted to place slowing towers where the creeps entered the green offense tower's range and red cleanup towers where the creeps exited the range. They also believed that, for slowing to work, you needed many slowing towers, more than would fit on the same column as the green offense tower (they would eventually use 33 towers). This necessitated placing towers on the walls opposite the green offense tower.

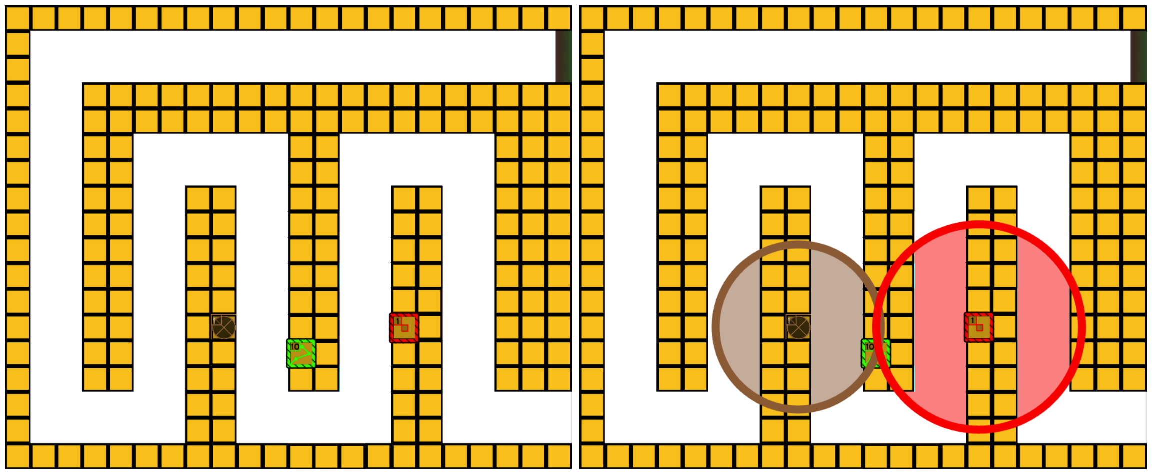


Figure 5.3. Subject E1500's Solution, Planning for Support Towers. LEFT: When placing the first tower, E1500 knows they will eventually place brown slowing towers near where the path enters the green offense tower's range and red offense towers where the path exits. Both will be on walls opposite the green tower. RIGHT: Both the brown slowing tower and red offense tower have two attack windows; the path passes by each tower twice, giving the towers two opportunities to attack.

Given this future development, the subject evaluated the positions opposite the green tower. On this map there are four columns, each identical in length and width to the others. The subject noticed that, while the columns were identical, the column's neighbors differed in one important opportunity. When the green tower is placed in either of the center two columns, the towers placed on the opposite columns would get two attack windows. In Figure 5.3, the path passes the brown tower once on the left side and then again on the right. The same is true for the red tower.

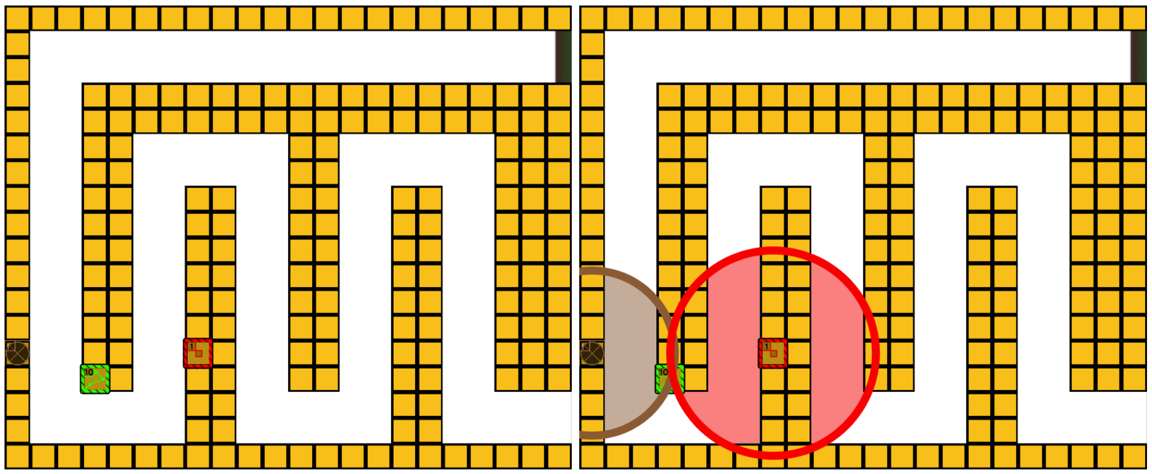


Figure 5.4. Effects of Shifting First Placement Left. If the green tower is placed on the first column, the creeps pass the brown tower only one time.

If the green tower had been placed on the first column (Figure 5.4), the brown tower would only get one attack window; it would attack half as often. If the green tower had been placed on the last column (Figure 5.5), the red tower would only get one attack window.

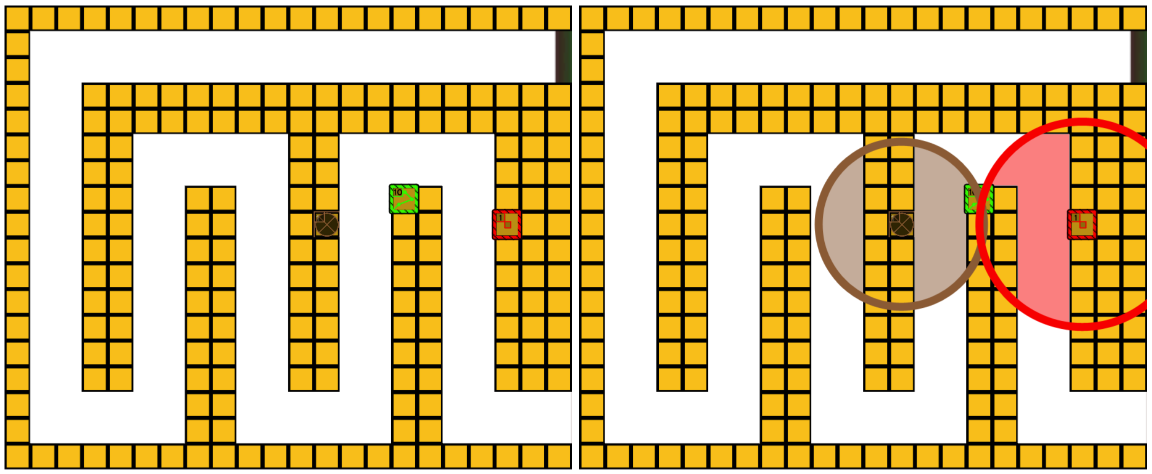


Figure 5.5. Effects of Shifting First Placement Right. If the green tower is placed on the last column, the creeps pass the red tower only one time.

Subject E1500 chose one tile back from the end of the U-turn on the 3rd column because U-turns allow towers to cover more of the path area, moving a tile back reduces wasted range and the 3rd column allows future, neighboring towers to also have good positions (i.e., positions that creeps pass twice).

Observation 3	Position of primary offense tower will determine how many attack windows adjacent towers will have.
Classification	Problem. Selecting certain positions will cause adjacent towers to lose attack windows.
Strategy 3	Place the primary offense tower where adjacent positions will receive multiple attack windows.

A fourth consideration is, presumably, that the offense tower needs wall positions nearby to place support towers on. This constraint was not mentioned by any subject on any map but we believe this is because there was no map on which this was an issue.

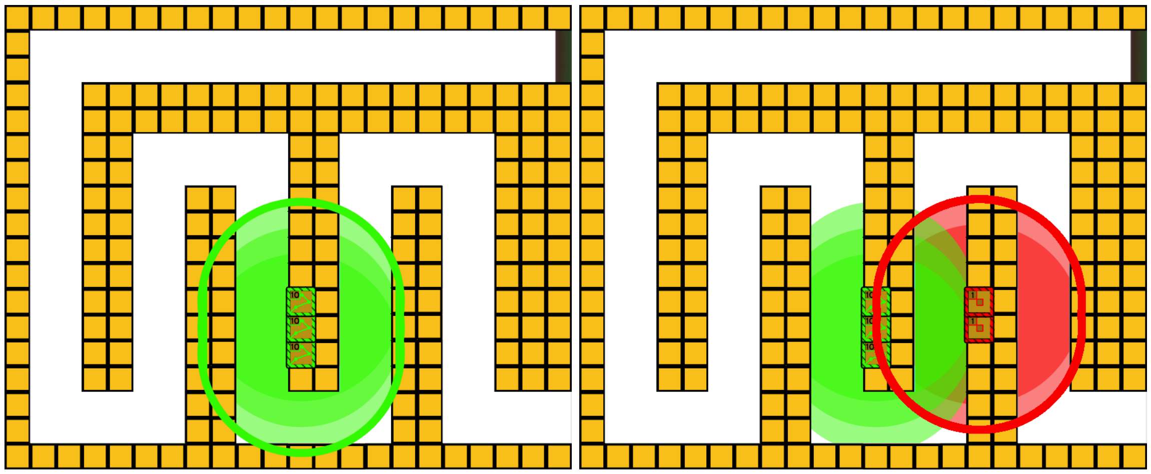


Figure 5.6. Subject E1500's Solution, Second Placement, Creating Offense Cluster. LEFT: Two more level 10 Green3 offense towers are placed next to the first tower. RIGHT: Two level 1 Red3 cleanup towers are placed at the end of the green tower's range.

Once the first primary offense tower was placed, the remaining primary offense towers were placed. During training, subject E1500 observed that scores were higher when the primary offense towers were near each other (we did not track training data and have no way to verify whether the data supported this belief). They therefore placed the two remaining primary offense towers next to the first one. When this happens, we refer to the first tower as the placement anchor. The following towers are placed relative to the anchor.

Subjects often explained why a certain observation was true but it is not clear whether this explanation created the behavior or was used to retroactively explain it. In one case, the subject explicitly stated they were not sure why towers placed close to each other earned a higher score but this did not stop them from using this information to form a strategy.

Observation 4	Scores are higher when the primary offense towers are close to each other.
Classification	Opportunity. Can obtain higher scores.
Strategy 4	Place primary attack towers close to each other.

The next towers the subject placed were two level 1 Red3 towers. These towers do a lot of damage but are slow-firing, unlike the Green3 towers which do significantly less damage but are 20 times faster (10 attacks per second vs. 0.5). Green3 towers also attack three targets at once; Red3 towers only attack one. The subject believed that, because of their speed, Red3 towers weren't suitable for being a primary attack tower but would be a good "finisher". Their high-level offense strategy was to use Green3 towers to weaken the creeps to the point that a single attack from a Red3 tower could destroy them.

The level 1 Red3 cleanup towers were placed so that the start of their range overlapped the end of the green towers' combined range. We refer to the first part of the range area of the Red3 towers as the placement target (the item to be placed). The three green towers form a kill zone. The anchor is the end of the range of the kill zone. The placement relationship is on, which is the same as overlap. We represent the placement decision as a 3-tuple of [target, relationship, anchor]. In this example this would be:

```
placement decision = [tower.range.front, on, killzone.range.back]
```

The result is shown in Figure E1500OffenseRange, right.

Observation 5	Red3 towers are too slow to destroy a large number of creeps.
Classification	Problem. Will let several creeps escape, lowering the score.
Strategy 5	Don't use Red3 towers as the primary attack tower.

Observation 6	Green3 towers are fast and powerful enough to wound all the creeps but not destroy them.
Classification	Problem. Will let several creeps escape, lowering the score.
Strategy 6	Use Green3 towers to reduce the number of creeps and wound the rest. Use Red3 towers to finish off the creeps that make it past the Green3s.

It should be mentioned that Observation 5 is not true and that what is written is a simplification of what subject E1500 said. They stated that Red3 towers are good for maps where the walls are skinny. To understand this comment it is important to know that Red3 towers have a much larger range than Green3 towers so when walls are thin, the Red3 tower's range is large enough to reach the other side of the wall and obtain another attack. In truth, even on maps with thick walls, a fully upgraded Red3 tower can reach most of the map. While they are slow, many subjects have successfully used them as the primary attack towers. In fact, subject E1500 is the only subject to use Red3 towers to support Green3 towers. The opposite (Green3 supporting Red3) is much more common.

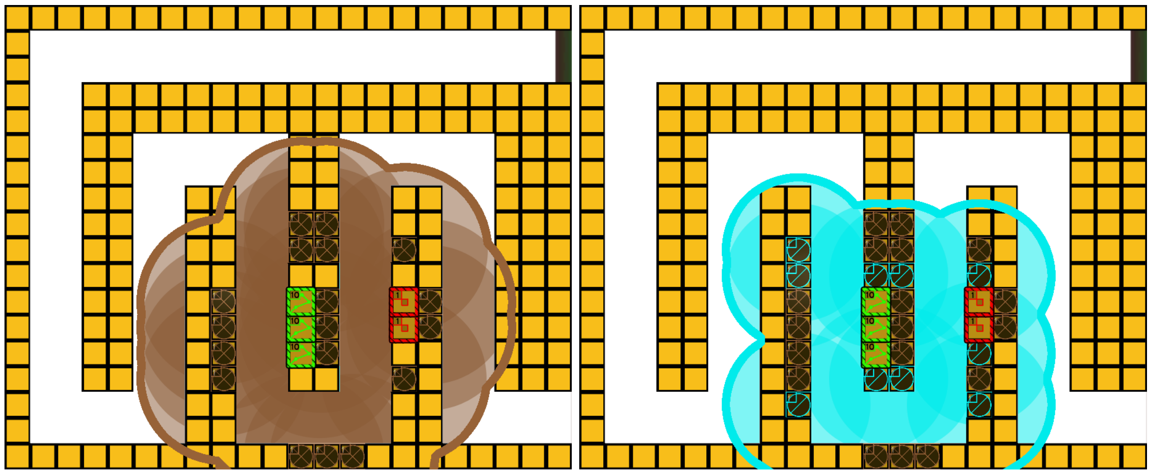


Figure 5.7. Subject E1500's Solution, Third Placement, Slowing Towers. LEFT: Placement of Brown freezing towers. RIGHT: Placement of Blue slowing towers.

Once the offense towers are placed, the subject places their slowing towers. Subject 1500 uses both Blue slowing towers, which cause four targets to move at half speed for two seconds, and the more expensive Brown freezing towers, which cause a single target to stop moving for two seconds. Many subjects used blue slowing towers exclusively for slowing. Others used a mixture of blue and brown towers, stating that they couldn't tell the difference between them in training and therefore decided to play it safe and use both. In contrast, subject E1500 has clear roles for the two different types of slowing towers.

Subject E1500 likes Brown freezing towers and considers them one of the keys to solving the puzzle. These towers have two problems, however. First, the subject didn't notice much benefit in training unless they used a lot of them. For this problem, the subject placed 18 Brown freezing towers around the kill zone and next to the red cleanup towers (Figure 5.7, left). Second, freezing towers can't hold a creep in place forever. They only stop a single creep and after the creep unfreezes it has a brief period of time to move before being re-frozen, allowing it to slowly move out of the freezing tower's range.

Blue slowing towers cannot prevent a creep from moving but a single slowing tower can slow down four creeps. Subject E1500 noted that Blue slowing towers are useful when used in combination with Brown freezing towers because slowing creeps gives the freezing towers time to re-freeze them. They estimated that one Blue and one Brown could hold a creep forever (we have not tested to see whether this is true). Accordingly, they placed 10 slowing towers around the kill zone (Figure 5.7, right).

Observation 7	Brown freezing towers stop creeps, giving the towers around them more time to attack.
Classification	Opportunity. Increases the time towers have to attack.
Strategy 7	Place Brown freezing towers next to the offense towers.

Observation 8	Brown freezing towers don't help much unless you use a lot of them.
Classification	Problem. Doesn't help much at low numbers.
Strategy 8	When using freezing towers, use large numbers of them.

Observation 9	Creeps eventually escape from Brown freezing towers because the creeps move a little before getting re-frozen.
Classification	Problem. Creeps eventually escape.
Strategy 9	Place Blue slowing towers next to Brown freezing towers to keep creeps from being able to move far while unfrozen.

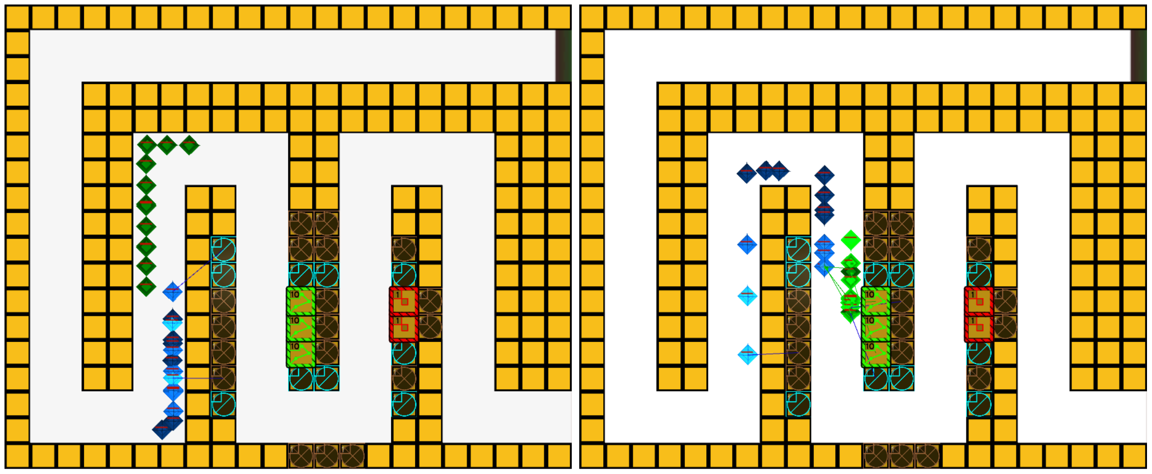


Figure 5.8. Execution of E1500's Solution, Differential Slowing. LEFT: Brown freezing towers hold back the right (blue) line of creeps, causing the two lines to separate. RIGHT: The last three creeps are held by freezing towers. They will never make it to the kill zone.

Slowing towers were placed across from the kill zone (the area covered by the Green3 offense towers) on the opposite column to slow down the creeps inside the kill zone. Although the slowing towers could reach both sides of the column, they were only placed on the kill zone side of it. The issue is not that the subject wanted to keep the slowing towers as close to the kill zone as possible (there are two Brown freezing towers placed behind the red cleanup towers at the end of the kill zone). The reason is that, when placed on the kill zone side of the column, the slowing towers' ranges can only reach one of the two lines of creeps on the non-kill zone side.

Figure 5.8 shows the slowing towers in action. The towers slow only the blue (right) line of creeps, not the green (left) line. This causes the two lines to separate. The green line goes into the kill zone alone, allowing the offense and slowing towers to focus entirely on that line of creeps. The blue line will arrive later, effectively giving the kill zone towers twice as many chances to attack the creeps. It also makes the Brown freezing towers and Red3 cleanup towers more effective as both are easily overwhelmed by large numbers of creeps that are concurrently within their range.

Observation 10	When placed at certain positions, slowing towers (both Blue and Brown) will slow one line of creeps and not the other. The non-slowed line will separate from the slowed one.
Classification	Opportunity. If the two lines of creeps separate and enter the kill zone one at a time, the kill zone towers can focus on a subset of the creeps, making them easier to stop and giving the towers twice as much time to stop the full group.
Strategy 10	Place slowing towers so that they slow only one line of creeps. Place these towers before the kill zone so that the lines of creeps enter the kill zone one at a time.

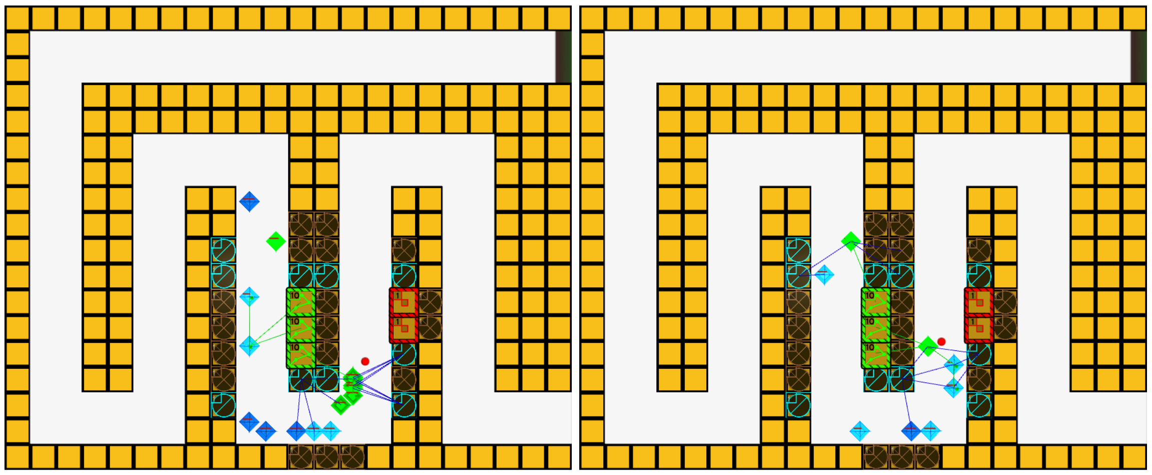


Figure 5.9. Execution of E1500's Solution, Kill Zone. LEFT: The inside (green) line of creeps is still separated from the outside (blue) line. The five remaining creeps on the inside line move into range of the red towers, which are able to focus entirely on the inside line of creeps. RIGHT: Only eight creeps remain. The 18 brown freezing towers are able to hold them in their positions permanently, giving the offense towers an infinite amount of time to finish them off.

Subject E1500's solution earned a perfect score of 28 (creeps stopped). They watched the execution of their solution and made several post-execution observations.

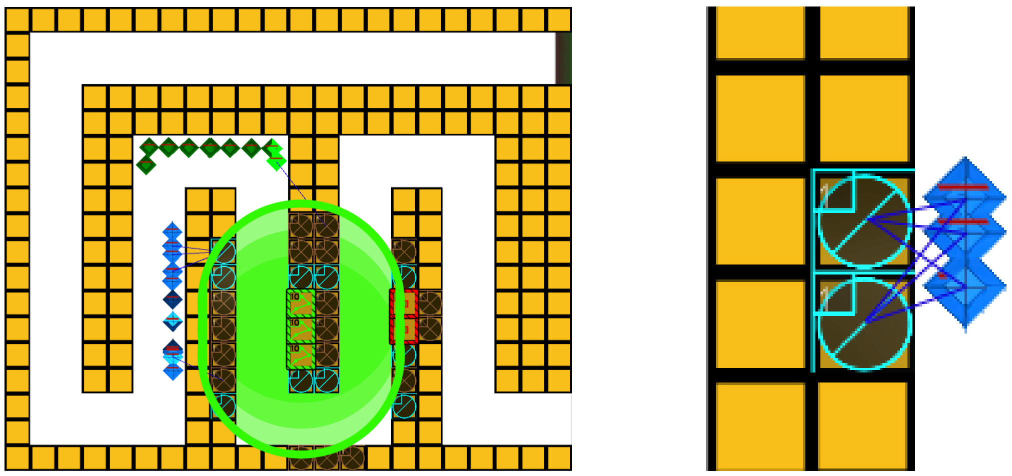


Figure 5.10. Subject E1500's Solution, Premature Slowing. LEFT: The slowing towers slow the outside (green) line of creeps before they enter the kill zone, allowing the inside (blue) line to shrink the gap between the two lines of creeps. RIGHT: When placed side by side, two blue slowing towers will attack the same targets.

Part of subject E1500's strategy was slowing one line of creeps so that the other could go into the kill zone alone. The subject noticed during execution that, because of where they had placed their slowing towers, the first line of creeps was slowed before they entered the kill zone, giving the second line time to catch up.

Observation 11	Slowing towers placed before the offense towers slowed the creeps before they entered the kill zone. The creeps moved slowly in an area where they couldn't be attacked by the offense towers.
Classification	Problem. Wastes slowing power.
Strategy 11	Make sure the entrance to the slowing towers' ranges is not in front of the start of the first kill zone offense tower's range.

Observation 12	Slowing towers placed before the offense towers slowed the first line of creeps before they entered the kill zone. This gave the second line of creeps some time to catch up, reducing the gap between the two lines.
Classification	Problem. Undoes the creep separation achieved by strategy 10.
Strategy 11	Make sure the entrance to the slowing towers' ranges is not in front of the start of the first kill zone offense tower's range.

Another thing the subject noticed was that, when two Blue slowing towers were placed next to each other they attacked the same set of creeps (Figure 5.10, right). This is because Blue slowing towers do not stop creeps, they just slow them down. They still move forward, entering the next Blue slowing tower's range and causing both towers to attack the same creeps. If a creep is slowed, being attacked by a second Blue slowing tower does not cause it to move even more slowly. The subject invented a feature to help describe this problem. *Diversity* describes how many towers are attacking the same creep. When many towers are attacking the same creep, diversity is low, which is bad. The problem could be solved, he determined, by placing Blue towers one tile apart from each other.

Observation 13	If two Blue towers are next to each other, they slow the same creeps.
Classification	Problem. The second tower is no longer slowing. Reduces diversity.
Strategy 12	Space the Blue towers out, leaving at least one tile between Blue towers.

It is worth noting that subject E1500 paid significantly more attention than most subjects to which towers were attacking which creeps as well as significantly more attention to the ordering of Brown and Blue towers, which many other subjects tended to place haphazardly. The subject stated that in the other tower defense games they had played, they could control which creeps each tower would attack. GopherTD lacked this feature, which bothered the subject, so they decided to arrange the towers in a way that would force the towers to attack the creeps they wanted. The majority of subjects did not pay close attention to which towers were attacking which creeps but the subject's experience with other tower defense games made them more sensitive to that feature, causing them to develop strategies based on that feature. Other subjects, not being aware of that feature, would not learn strategies based on manipulating it. This is an example of how perception controls the features the subject recognizes which controls the goals they form which control the strategies they develop.

Observation 14	You cannot explicitly control which creeps a tower attacks.
Classification	Problem. Many past strategies depend on this feature. Decreases diversity.
Strategy 13	Manipulate the order and spacing of towers to control which creeps a tower attacks.

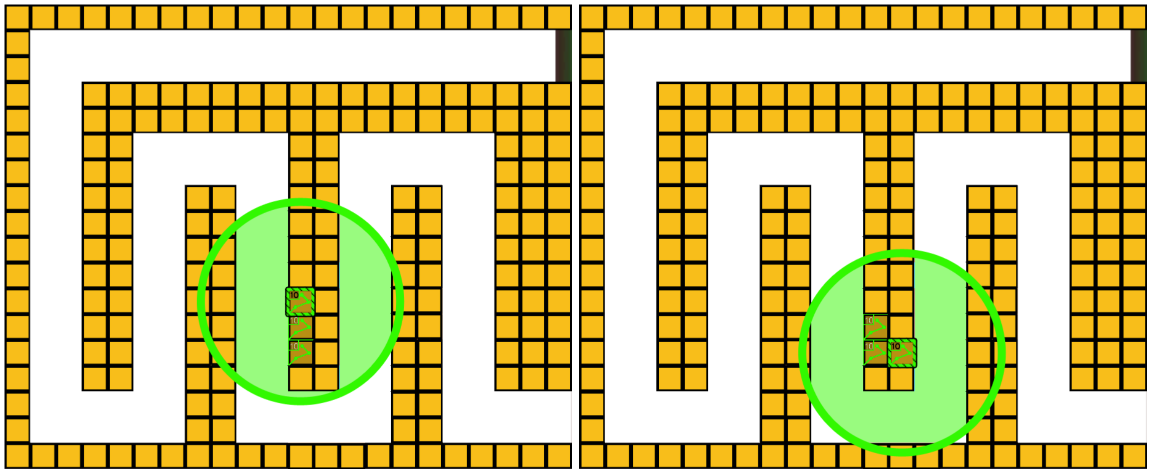


Figure 5.11. Subject E1500's Solution, Offense Placement Mistake. LEFT: Range of top Green3 tower. RIGHT: Range of Green3 tower if it were moved down.

The subject's final observation was that they had placed one of the Green3 primary offense towers badly. They had started one space away from the end of a U-turn and then built a chain from there. This led to the last tower not covering the end of the U-turn (Figure 5.11, left), meaning it did not cover as much area as the other green offense towers. The tower would have been better placed to the side of the first tower (Figure 5.11, right).

It is not clear what caused the subject to notice this. It would be hard to notice during the execution of their solution as the range circles do not typically show during execution.

Observation 15	A chain of towers placed on a U-turn result in some towers not having the advantages afforded by the U-turn.
Classification	Problem. Tower covers less range and therefore attacks less often.
Strategy 14	Group towers next to each other, as if in a ball.

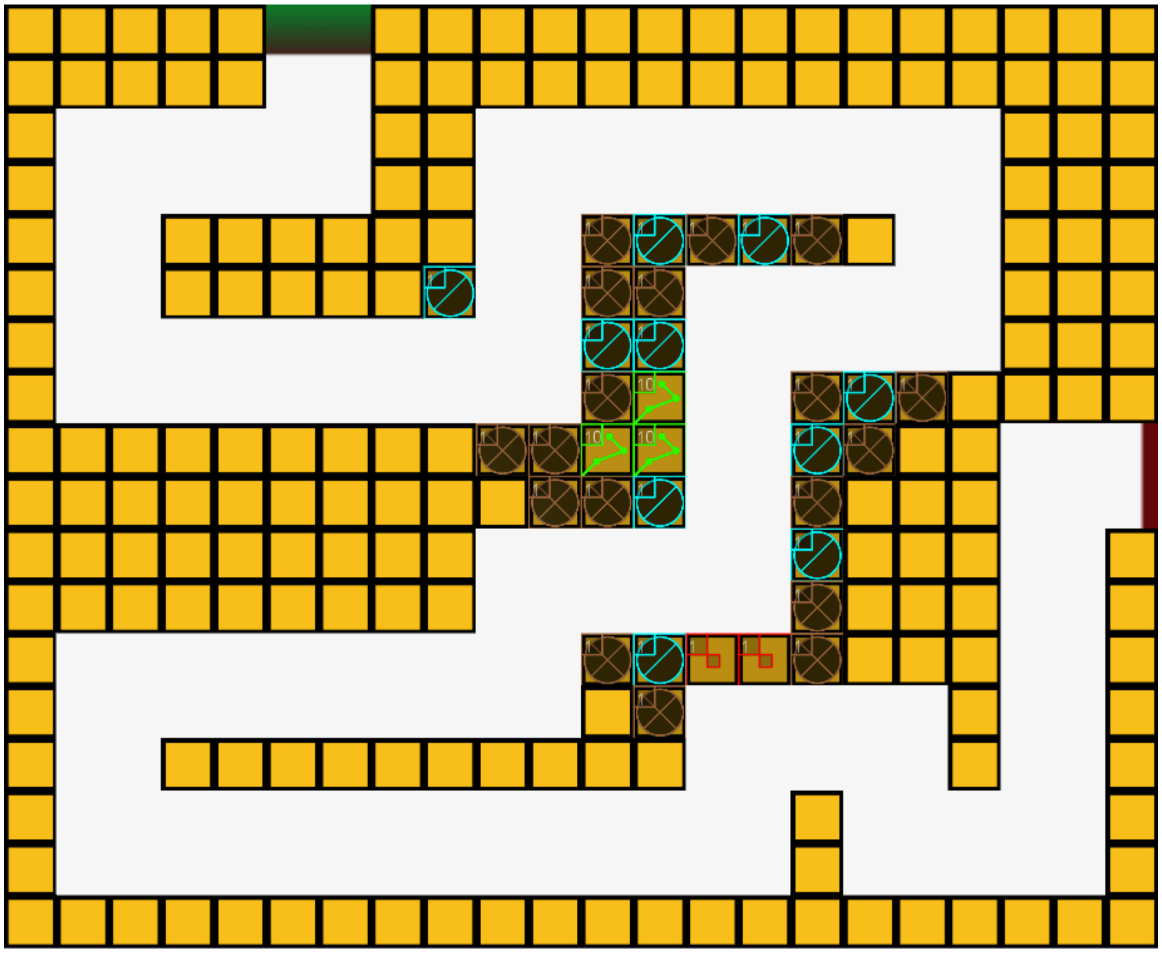


Figure 5.12. Subject E1500 Solution for Snaking Path.

After the map Up and Down, the subject attempted to solve the map Snaking Path. Figure 5.12 shows both how the subject's strategies are applied to a new problem and their post-execution observations. There are once again three level 10 Green3 towers surrounded by a mixture of Blue and Brown slowing towers. Two level 1 Red3 towers are placed at the end of the kill zone. The produced solution is in no way similar to the last solution structurally but, when viewed from the point of view of the goals and strategies, is quite similar. The two solutions are not similar in themselves but, to one who knows which strategies exist in this domain, they are clearly similar in the processes and more abstract goal- and affordance-based considerations that created them.

There are two other important things to notice in this solution. First, some of the strategies used on the last problem were not used here. As an example, strategy 10, slow one line and not the other, was not used. This is because this map lacks the necessary affordances; there are no places on this map where a tower can both slow one line but not the other and be used to slow creeps in the kill zone. The subject also did not use strategy 1, use U-turns. When building their solution, they initially placed their first tower on the U-turn. Once they had done that, they noticed that, because the U-turn was skinnier, the range of their level 10 Green3 tower went into walls on all three sides. Since this was deemed wasteful and could not be solved in the same way as last time (strategy 2, move the tower back one tile), they moved the tower to a place where that problem didn't exist, ruling out all of the U-turns present.

The second thing to notice is the influence of the post-execution observations from the last problem. Strategy 12, space out blue towers, is used, as also is strategy 14, group the green towers together in a ball.

5.2 Representation: Description vs. Problem Solving

Much has been written about spatial reasoning. Many of the basic tasks are validation problems – if we know that A is behind B and B is behind C, is it valid to say that C is behind A? This can be extended to constraint satisfaction problems – given a set of variables, relationships between the variables and values to plug into the variables, is there a set of variable assignments that does not violate any constraints? (Lee and Wolter 2011)

At the application level, much of the existing work focuses on identifying items in a scene. Examples include identifying objects (Hué et al. 2011), performing GIS queries to identify an area (Duan et al. 2013), sketch recognition (Hammond and Davis 2006; Hammond 2007), solving spatial analogy problems (Tomai, Forbus, and Usher 2004;

Lovett, Forbus, and Usher 2007) and corridor identification for path planning (Forbus, Mahoney, and Dill 2002). Much of the remaining work focuses on determining how actions in a spatial setting will affect that setting. Examples include inferring the effect of turning gears in a clock (Forbus, Nielsen, and Faltings 1991) and determining how to throw a ball such that it lands in a basket (Kreutzmann and Wolter 2011).

While these examples encompass a broad range of tasks, they have many things in common. The goal is spatial in nature – the task is either to classify data in the scene or to predict changes to the scene. The reasoning uses spatial features and relationships that can be defined independently of context or goal. For example, a circle is a circle regardless of whether it is being used as a gear in a clock or wheels on a car. The definition of corner does not depend on whether the reasoner's goal is to predict the movement of a machine or identify a partially obscured object. The features used are physical objects – points, polygons and other observable data. Relatively little work has been done on the integration of temporal and spatial reasoning, although benchmark problems motivating work in this area have been recently proposed by (Kreutzmann and Wolter 2011). The important thing to recognize is that the current work in spatial reasoning is limited to spatial goals and problems. It might serve as a pre-processing step to a non-spatial task (e.g., validating bridge design (Valentine et al. 2012)) but the spatial reasoning is not integrated into non-spatial problem solving.

The task presented in this thesis, solving tower defense puzzles, also involves spatial reasoning. All of the actions are spatial – the only action the problem solver can take is to place towers at locations on a map. The reasoning task is to decide where on the map to place the towers, a spatial decision. Where this work differs from traditional spatial reasoning work is that the goal in the tower defense task is non-spatial. The problem solver tries to maximize their score, a non-spatial quantity, by stopping as many objects as possible from reaching a designated point on a map. As such, the tower defense task differs from the types of problems currently being investigated.

The distinction might seem unimportant or simply a matter of semantics. After all, even if the goal is non-spatial, all of the reasoning is about spatial actions in a spatial domain. The goal is affected solely by the manipulation of space. How much do things change if the goal is not explicitly spatial?

The answer is, a lot. The work discussed here required a radically different method of representing space (or, more appropriately, the problem) and reasoning about it. In this chapter we do away with spatial features and introduce goal-directed spatio-temporal representations known as *affordances* (Gibson 1979) and show how the definition of these affordances depends on the problem solver's goals (i.e., a circle isn't always a circle) and how it is representation, not reasoning, that is responsible for the intelligence of the system. In this chapter we also show that spatial reasoning does not use, much less require, spatial data. Much of the work on spatial reasoning uses complex techniques such as constraint satisfaction algorithms and algebraic geometry. In this chapter we show how these problems can be efficiently solved using sets of simple, single-purpose, affordance-based strategies. Along the way, we will touch on how a problem solving approach based on affordance-based representations and sets of simple strategies supports learning from experience and transfer learning.

5.3 Components of a Solution Attempt

Each tower in a solution attempt belongs to a role and functional group, is placed according to one or more placement decisions and is placed across the map according to a distribution.

Each of these towers has a purpose, a role to play in the overall solution. We refer to these as the object's role. Common roles include primary offense, supporting offense, slowing and cleanup. The primary offense towers attempt to destroy the creeps.

Supporting offense towers are less powerful but also significantly less expensive. They are located next to the primary offense towers and help destroy the creeps. The slowing towers (technically, SLOW IN RANGE towers; see Chapter 8 Strategies for more examples of slowing tower roles) slow down the creeps while they are in range of the offense towers, giving the offense towers more time to attack. These towers (primary offense, supporting offense and slowing) are often grouped into one of more kill zones. Done well, these should destroy most of the creeps but it is not uncommon for a few creeps to make it through. Cleanup towers are placed after the kill zone and are used to catch these creeps. Figure 5.13 gives an example of tower roles.

Each role has one or more functional groups. All towers in a functional group are placed in the same area according to the same placement strategies. The reason there can be more than one functional group is because multiple towers can have the same role but be placed in significantly different places. This most commonly occurred in the experiments when a subject wanted to create multiple kill zones (e.g., one at the start of the map, one in middle and one near the end). For the towers in the primary offense role, the subset in each kill zone would belong to the same functional group. In Figure 5.13, each role has only one functional group. See Chapter 7 Solution Representation for examples of roles with multiple functional groups.

Placement strategies are used to determine placement decisions for each functional group. A placement decision is a functional description based on affordances. It does not state where a tower is placed, it describes a goal. For example, instead of

> "Place a level 10 Green3 tower at position (12, 6)",

a placement decision might say

> "Place a fast, heavy offense tower such that the place where the creeps enter the tower's range is aligned with the path entry of a slowing tower".

Because they are stated in terms of desired outcome, placement decisions are relatively straight forward to map to placement strategies.

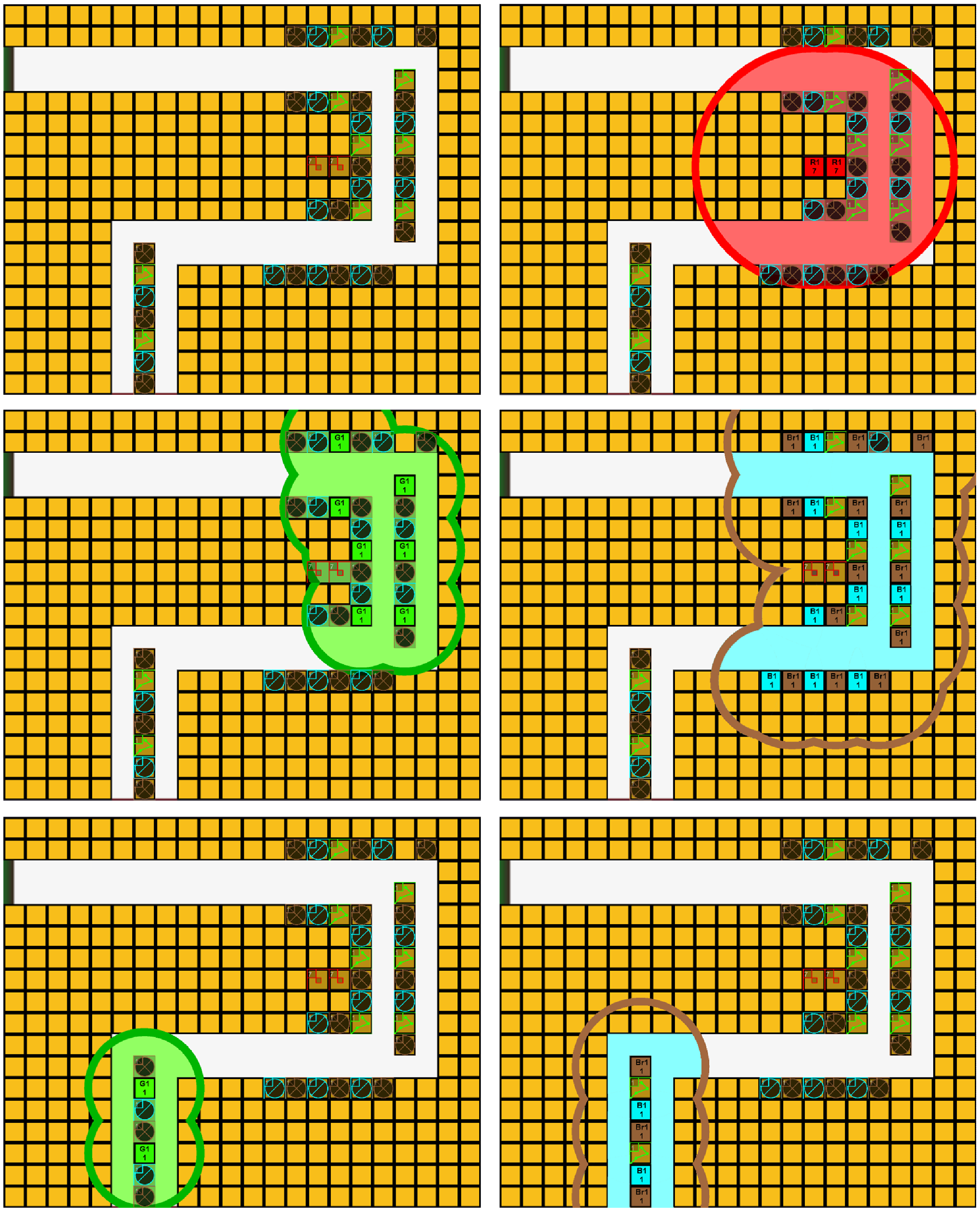


Figure 5.13. Subject e2100's Solution to the Map Pathways. TOP LEFT: The solution. TOP RIGHT: The primary offense towers. CENTER LEFT: Support offense towers. CENTER RIGHT: Slowing towers supporting the offense towers. BOTTOM LEFT: Offense towers in the cleanup zone. BOTTOM RIGHT: Slowing towers in the cleanup zone.

A placement decision has three parts, a placement target, placement relationship and placement anchor. The above example would be represented as

```
[target=Range Entrance,
 relationship=On,
 anchor=Range Entrance of a tower with role=Slow In Range]
```

The placement target is what is being placed. It is not necessarily a tower. In the preceding example, the agent is attempting to place a tower's range's entrance at a specific point. This obviously dictates where the tower must be placed but it is not the tower that is being reasoned about.

A placement target is placed in relation to a placement anchor. The placement anchor is a reference point. There are a number of anchors, from towers to map properties to ordinal quantities. We observed several dozen different types of anchors, not counting compound anchors. These are discussed in Chapter 7 Solution Representation.

The placement relationship specifies the relationship between the target and anchor. Values can be absolute (on, intersecting) or relative (before, long after), precise (touching) or vague (near).

A placement decision does not necessarily specify a specific position. It specifies criteria that multiple positions might match. A strategy will often specify a location such as "along the path and near the group" that can be described by a set of simple placement decisions. In this example, the decisions would be

```
[tower, touching, path]
[tower, as close as possible, tower group]
```

Each placement decision results in a set of positions and the desired position is the union of those sets.

In Figure 5.13, the primary offense towers are placed where their range covers the largest number of path tiles. The support offense towers are placed near the primary offense towers. The primary slowing towers slow the path area covered by the primary offense towers. The cleanup offense towers are placed near the exit. The cleanup slowing towers slow the path area covered by the cleanup offense towers. More formally:

Role=Primary Offense, Functional Group=1

```
[range, maximize, usable range]
[tower, near, group 1]
```

Role=Supporting Offense, Functional Group=2

```
[range, maximize, usable range]
[tower, near, group 1]
```

Role=Slow In Range, Functional Group=3

```
[range, intersect, usable range of group 1 not already slowed]
[range, maximize, usable range]
```

Role=Cleanup Offense, Functional Group=4

```
[range, near, map exit]
[range, maximize, usable range]
[tower, as close as possible, group 4]
```

Role=Cleanup Slowing, Functional Group=5

```
[range, intersect, usable range of group 4 not already slowed]
[range, maximize, usable range]
```

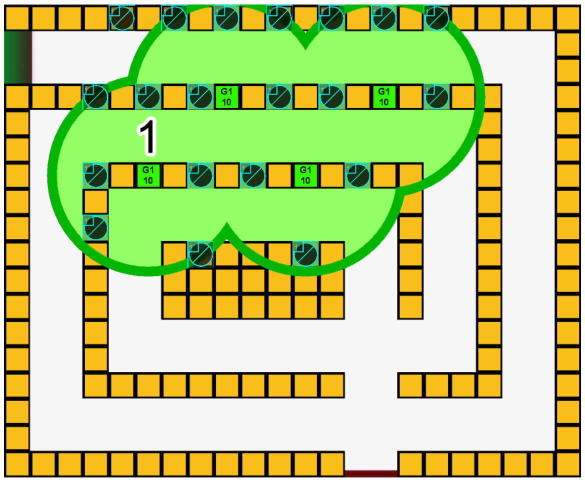
It is worth noting that the order of placement is important. In this example, group IDs reflect the order that each group is placed.

The final element of a solution description is the distribution. Many subjects had no preference for how their towers were distributed but some strongly preferred concentrating their towers while others spread them across a region or the map. A common distribution pattern was chaining – placing towers in some form of line. Chains could be concentrated, spaced out or distributed over a range. Some subjects chose to use a pair of staggered chains, forming a pattern similar to teeth in a zipper. A full discussion of tower distribution and chaining is given in 7.5 Tower Distribution.

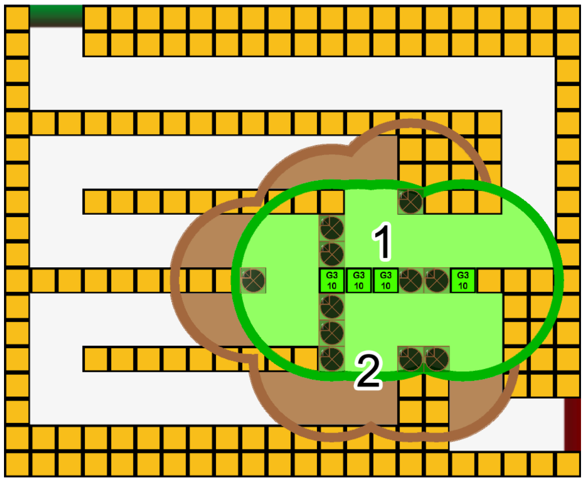
5.4 Reasoning by Strategy

In this section we talk about *reasoning by strategy*, a topic that has been studied in humans, both in the psychological literature for expertise and bounded rationality, but not much in computer science.

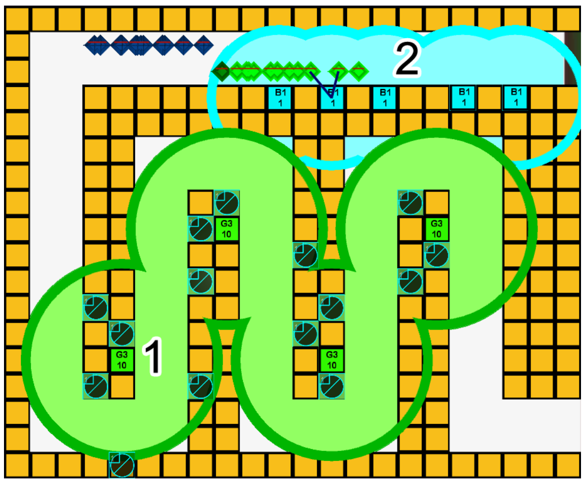
What do we mean by reasoning by strategy? In this work we define the term to cover the type of reasoning we observed in our subjects. Before we attempt a definition, consider the following examples. What follows are excerpts of three subjects explaining, in their own words, how they solved a specific puzzle. The last subject, 3E0200, is included twice to show how a strategy he learns on one puzzle is carried forward to another. Quotes are taken verbatim from think aloud data and are therefore rather informal and prone to run-on sentences.



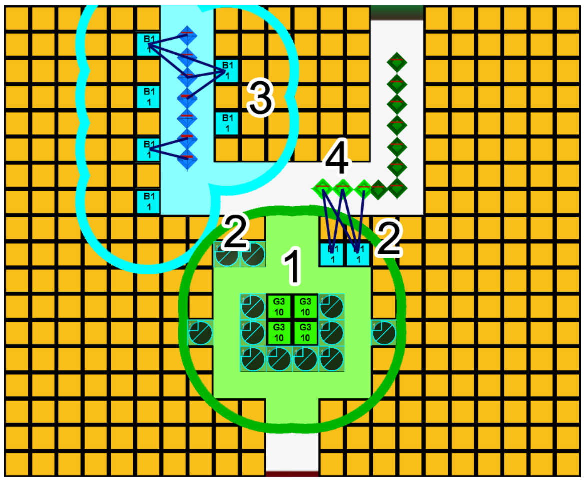
Place [the green offense towers] here [1] so that the creeps go by them more than one time so [the creeps] get hit more often by each tower. And then use the blue ones and put them near all the green towers to slow them down on both sides. That way they're in range of the green ones for as long as possible. Then kind of spread them out to give them room. – 3N0100



I think I'm going to focus a lot of my power in this area [1] because [the outer line of creeps will fall behind] so it's the time difference again. So I'm going to put a couple of [green offense] towers right there... Another level 10 right there, [that] seems right… I'll put a couple [of slowing] towers in… there [2] so that it slows down right around the [green offense towers] so the greens have the most time to fire because they don't have much range which is kind of annoying but they shoot really fast… I don't usually use green towers because they annoy me but they seem… like they'd work better [here] because it's an enclosed space, they have a lot of different chances to fire at the enemy. – E0600



One of the things I found was that, in some levels, depending on where the units spawn, you'd be able to scoop them up based on where you put the [slowing] towers… We couldn't control what the towers [1] were aiming at and it took a little bit of… manipulation to get the towers to shoot at the same target and delaying one wave on one side of the map [2] to get the other wave to go straight worked pretty well on some of the levels. – 3E0200



I do the thing where I pull one side using the blue towers and then catch everything with the green towers in the middle. And I found it worked better to put the green towers here [1] than it did if I put them up here on the left and right side [2] and split them up and I think it's because they focus [on] the same creeps each time instead of different ones. So I add slows here [3]. Going to pull the left side to go slower than the other one. [During execution] It looks like I slowed the green creeps [4] too which is going to be a problem. Probably would have been better if I had left the ones on the right side out. – 3E0200

There are several things to notice.

We argue that a strategy leads to a single decision. While it could be argued that everything each subject describes is part of a single, multi-part strategy, we feel it is more natural to consider each subject's approach as a set of simple, single-purpose actions. This has several benefits. Consider an agent with a set of strategies that uses a subset of those strategies to solve a specific problem. Upon encountering a new problem, the agent can apply one of those strategies to the problem without being required to use the others, which might not apply to the new problem. Novel problems can be solved by using novel combinations of existing strategies. If a person's problem solving approach is a combination of strategies, combinatorics allows for each person to have a unique approach or style without requiring an unrealistically large number of unique strategies. A strategy, in this case as a simple placement decision, can be a learning target and thus is learnable, either inferentially or through experience. Individual strategies can be added to an agent's strategy set, allowing them to improve with experience or education. Support for each of these topics is offered throughout this chapter. Chapters 6 and 7 provide detailed information on strategies and representations respectively and help illustrate the concept of problem solving as a large set of simple strategies.

Strategies have goals. The goals are simple, objective and measurable. The goals are tightly tied to an action – little deliberation is necessary. For example, subject 3N0100 states the goal to have creeps be attacked more often by placing the towers at locations that the creeps pass by twice. This is done by finding positions where the path passes two or more times and placing the tower there. Subject E0600 states the goal to have the offense towers have more time to fire by slowing the creeps that are in the range of those towers. This is done by placing a slowing tower next to the offense towers. In each case, the goals make it trivial to determine the action.

Strategies have goals that are responses to problems and opportunities. Subject E0600 wishes for their offense towers to have more time to fire because, in their opinion, the green offense towers they'd chosen "don't have much range". This opinion was formed from the observation that, when they failed to solve a map, it was because the creeps escaped, which the subject attributed to the towers not having enough time to do their job. Subject 3E0200's goal of slowing one line of creeps so that it fell behind the other was based on their concern that, when both lines were present, their group of offense towers failed to agree on which creep to focus on. The last example shows a subject in the process of developing new, additional strategies to supplement their existing ones by noticing execution failures (right line of creeps is slowed before they can be attacked, partially undoing the effects of slowing the left line) and attributing a cause to them (placed slowing towers in a place where it slowed the wrong line too early).

Variables in strategies are moderately abstract. This is difficult to see in the quotes above. Subjects frequently talk about placing blues and greens, terms that are fairly concrete. If this is how subjects reason about the problem, the knowledge could not be transferred to other domains, even to other tower defense domains (green and blue towers with the specific properties of these are specific to GopherTD). In interviews it was discovered that, while subjects talked about concrete objects, they reasoned in terms of capabilities.

In their minds, they were thinking of fast offense towers that hurt creeps (greens) and slowing towers that temporarily slowed them down (blues). The majority of tower defense games have towers of this type, although specific properties differ (i.e., they might do more or less damage, cover more or less range, be more or less expensive). Many subjects said they were using variations of strategies they had used in other games. They began treating the GopherTD towers as if they were the same as the towers in other games they knew and then, based on experience, refined their opinion of the towers' capabilities. Often strategies stayed intact at the strategy level but differed in execution (e.g., using more towers in GopherTD if the game they already knew had more powerful but expensive slowing towers). We noticed that this transfer was not always beneficial. Negative transfer was most common among those subjects who had experience with tower defense games where slowing towers were less powerful than other options, causing them to rule out strategies in GopherTD that relied on slowing towers.

While variables were more abstract than the concrete objects available in a specific domain, we saw no evidence that they were stored any more abstractly than "fast attack tower" and "slowing tower". These concepts are not particularly abstract. They are tied to the set of domains that have objects similar to attack objects placed in a spatial environment for the purpose of stopping other objects (we believe the concept of slowing tower applies to fewer domains than attack towers). This extends to other tower defense games, real-time strategy games and military operations. The equivalent of an offense tower might be a guard, tank, speed trap or salesman. While near transfer appears to be possible through reasoning by strategy, it seems difficult to see how the concept of fast attack tower could be abstracted to the point of, say, helping a surgeon determine how to operate on a tumor (Duncker and Lees 1945). This is not surprising given that numerous studies have shown that far transfer rarely happens (Lave 1988; Singley and Anderson 1989; Detterman and Sternberg 1993). It is possible, however, that while processes and relationships don't transfer, the way data is represented does (Novick 1990).

Subjects used strategies for numerous tasks including selecting towers, placing towers, selecting strategies, combining strategies and learning strategies, both actively and passively. Given the breadth in each area, we limit ourselves in this document to discussing only strategies for placement decisions. Chapter 8 Strategies discusses 80 placement strategies observed across six categories:

- **Spatial Placement**. Strategies for placing towers that rely predominantly on spatial affordances.
- **Temporal Placement**. Strategies for placing towers that rely predominantly on temporal affordances.
- **Satisficing**. Strategies used by subjects to minimize measurement, cognitive effort or demands on working memory.
- **Targeting**. Strategies for placing towers in such a way that they select desired targets.
- **Efficiency**. Strategies for placing towers in a way that conserves resources.
- **Polish**. Strategies for improving solutions.

Placement strategies can be defined as one or more placement decisions. Consider 3N0100's strategy "use the [slowing towers] and put them near all the [offense] towers to slow them down on both sides. That way they're in range of the green ones for as long as possible." There are many ways to achieve this so let us assume we wish for the slowing tower to be near the offense tower but also adjacent to the path since slowing towers have short ranges and the closest position to the offense tower might be in the center of a wall, preventing the tower from being effective. We can represent this strategy as the pair of placement decisions:

```
[tower, touching, path]
[tower, near, group(offense tower)]
```

Alternatively, if we wish to distribute the slowing towers so that a set of slowing towers covers all of the usable path area that the offense towers cover, we could represent the strategy as:

```
      [tower, touching, path]
      [range, inside, usable range of group(offense tower) not slowed]
```

This version is more complex because it requires the towers to be placed such that the path area they cover is in range of an offense tower but not of another slowing tower. Other complex strategies involve placing towers at places where the two lines of creeps have been separated, where paths overlap, in a chain, in the space between two kill zones, after the creeps have passed through all kill zones and placing all towers in a single wall region.

Subjects routinely used strategies such as these without ever complaining about difficulty in executing them. This is because the majority of the work is done by the perceptual system. Once the perceptual system has identified the relevant affordances, a complex strategy such as placing a tower on a wall opposite of the region containing the offensive core (i.e., the location that has the largest number of offense towers) is simply "place X on Y".

Once the relevant affordances have been identified by the perceptual system, strategies in this domain are roughly as simple as the ones observed in other domains. Solving tower defense puzzles is complicated by two issues not present in other domains described in the bounded rationality literature. First, other domains studied involved a single decision. Our domain involves solving complex problems involving numerous decisions, all of which must work together synergistically.

The second issue is that of the perceptual system – by leveraging their innate perceptual system, it is often trivial for humans to recognize the required affordances (although it is often a conscious decision; subjects that did not use a given affordance in their strategies were unlikely to recognize it in a problem). Computers do not have anything equivalent. From this it can be argued that the core impediment to achieving strong spatio-temporal problem solving abilities in computers is not a deficit in reasoning or representation but the lack of a perceptual system. We argue, however, that reasoning and representation only work when they are in harmony with the perceptual system. This requires a move from features to affordances. Affordances are discussed in the next section.

5.5 Spatial Representation

5.5.1 Features, Properties and Relationships vs. Affordances

When doing human experiments, where a human subject explains a concept to a human experimenter, it is easy to overestimate one's understanding of the concept. Some concepts seem so simple and obvious that it is hard to imagine that they could be misunderstood. We felt that many basic spatial concepts fell into this category. How could anyone misunderstand the concept "U-turn"? The term "near" is vague in its limits but when given two positions, there is no ambiguity about which is nearer some reference point. Our attempt to convert these well understood concepts into a computer model showed us how very wrong we were.

Consider the case of subject E0100, whose strategy is to place heavy offense towers near the end of a U-turn and then surround the U-turn with slowing towers (Figure 5.14). Our initial attempt at creating an agent to mimic E0100 consistently and disastrously failed to replicate the subject's solutions. The problem had to do with differences in what was meant by "U-turn".

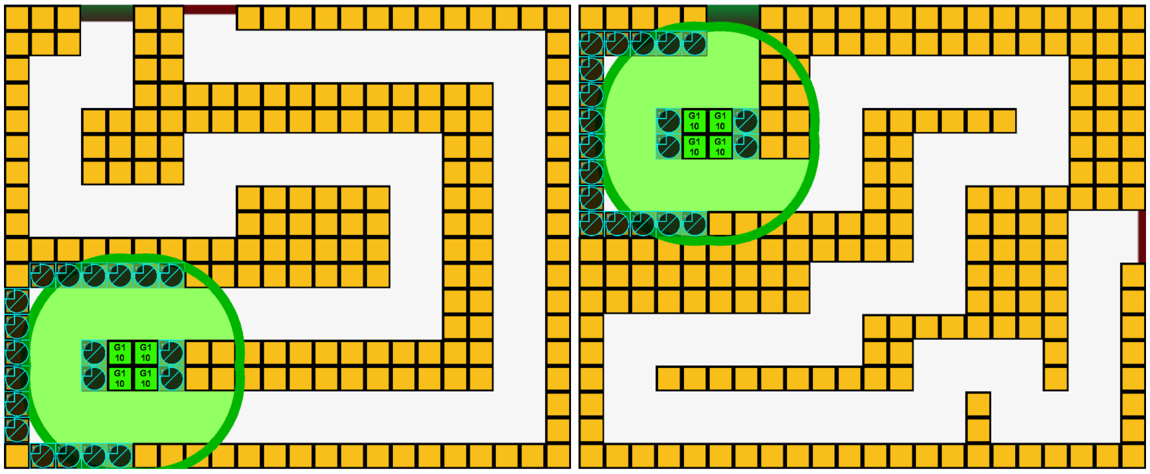


Figure 5.14. Subject E0100's Solution. LEFT: Subject E0100's strategy for the map Elemental-ish was to place four level 10 Green3 heavy offense towers near the end of a U-turn and then surround them with level 1 Blue1 slowing towers. Green area shows the range of the offense towers. RIGHT: E0100's solution for the map Snaking Path is similar in structure to their solution to Elemental-ish.

When we created a computer definition of U-turn, we began by looking at the geometry – a U-turn had three sides so the end of any rectangle was a U-turn. We quickly realized this wasn't suitable for the domain. Relative to the goal of the game (stopping objects that are following a path), a U-turn is defined by the direction of movement – there need to be three contiguous path segments with the outer two moving in different directions. This definition of a U-turn is simple, objective and domain-independent. It is size and rotation invariant. U-turns defined in this way are easy for the computer to identify from directional path data. To our mind, our spatial parsing system using this definition did a great job identifying U-turns. For example, on the map Elemental-ish, the computer identified seven U-turns (Figure 5.15).

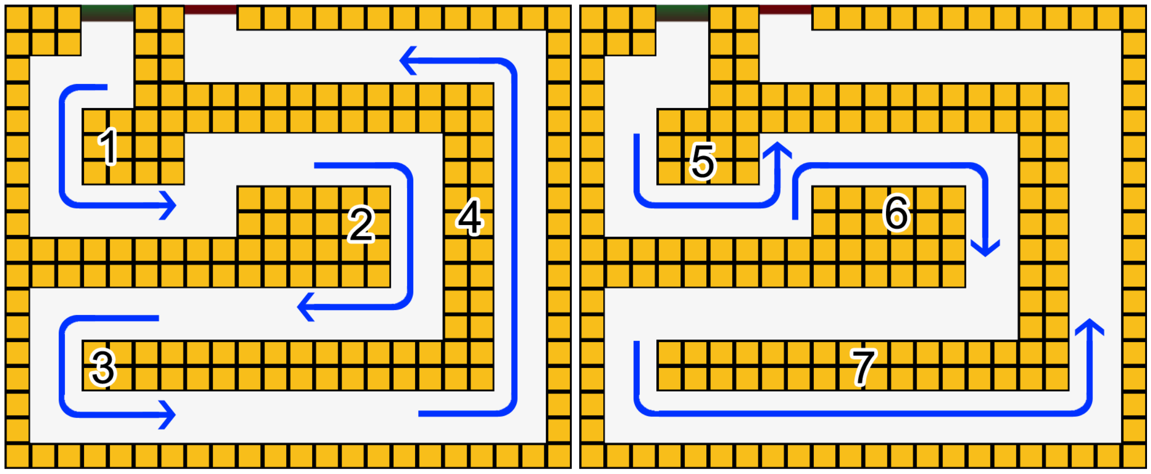


Figure 5.15. Objective U-Turns. The seven directional U-turns on the map Elemental-ish. LEFT: Vertical U-turns. RIGHT: Horizontal U-turns.

The agent using this definition of U-turn was a disaster. Our mistake was in believing that, to our subjects, spatial features had objective definitions (i.e., a circle is a circle independent of who is asking or why they're interested). Although obvious in retrospect, when subjects reported their strategies in terms of spatial features (U-turns, corners, passes, islands, areas, corridors, etc.), what they actually meant is that their strategies depended on abstract, goal-relevant properties for which spatial terminology was a short cut both for explaining their reasoning and searching for relevant areas on the map.

It is important to remember that our subjects were not interested in representing space, they were interested in solving problems. They represented space only because it furthered that goal. As such, their representations of space were largely limited to those that helped solve problems. This means that they not only avoided representing spatial features that were not useful to problem solving, the features they did represent were represented differently, with the meaning of terms often changing depending on context and intent.

For subject E0100, a U-turn allowed them to place four level 10 Green3 towers in a concentrated area and forced both lines of creeps to move through those towers' range on three sides. Although the map Elemental-ish contains seven U-turns, only one of them was large enough to allow the subject's preferred solution (a 4x2 arrangement of blue and green towers surrounded by blue towers) and small enough that the green towers' ranges could reach all three sides. The other six U-turns were unusable by E0100's strategy (Figure 5.16).

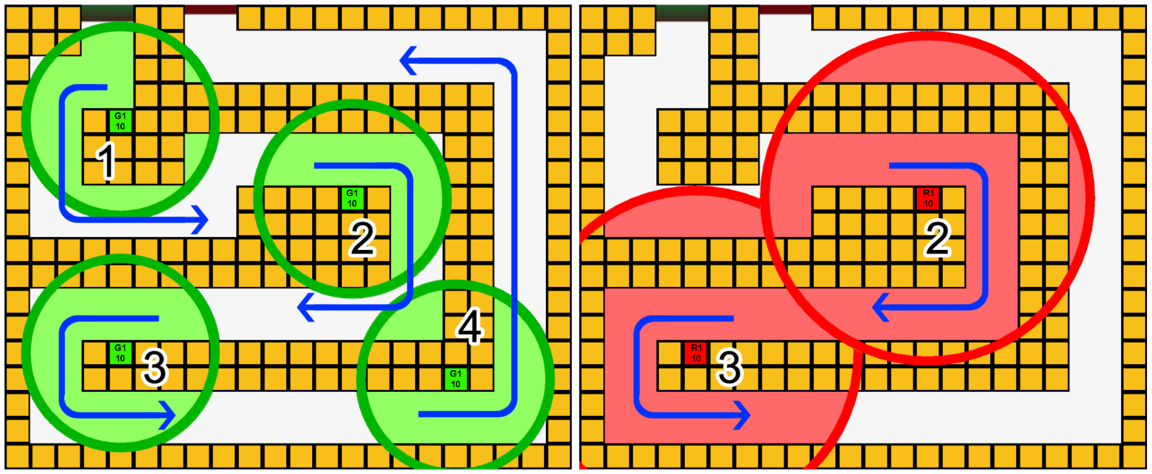


Figure 5.16. Usability of U-Turns. LEFT: As far as a level 10 Green3 tower is concerned, this map has only one U-turn. The remaining six are too wide for the tower to reach all three sides for both lines of creeps. RIGHT: Red towers have a larger range than green towers. As far as a level 10 Red3 tower is concerned, this map has five U-turns. Only the widest two (bottom and far right) are too wide for the tower to reach all three sides. Technically, the center U-turn is an island, not a U-turn, since the tower can see four sides.

In goal-directed spatial reasoning, the subject is not interested in spatial features but spatial affordances – the opportunities a spatial arrangement offers for a specific objective. The opportunities afforded by a spatial arrangement depend on the tool being used and the actions the agent knows about. In this case, the tool is a tower. If the definition of the spatial affordance U-turn depends on the tower's having a range large enough to see three contiguous path segments heading in different directions, the number of U-turns on a map is contextually dependent on the tower being placed. From the point of view of a level 10 Green3 tower, the map Elemental-ish has a single U-turn (Figure 5.16, left). From the point of view of a level 10 Red3 tower, which has a much larger range, the map has five U-turns (Figure 5.16, right). Technically, for the red tower, one of the U-turns is an island – for the green tower, the center area is a corner because the tower can only see two contiguous path segments while for the red tower, it is an island because the tower can see four. In spatial affordances, the definition of a spatial feature is relative to the capabilities of the goal seeker.

An unexpected source of confusion was that subjects used terms in what appeared at times to be completely incorrect ways. Consider the strategy, "place a slowing tower near the offense tower" and the positions in Figure 5.17. Position P1 is clearly closer to the offense tower at position A than position P2 is. Even so, it is obvious to the subject that their strategy means to place the slowing tower at position P2.

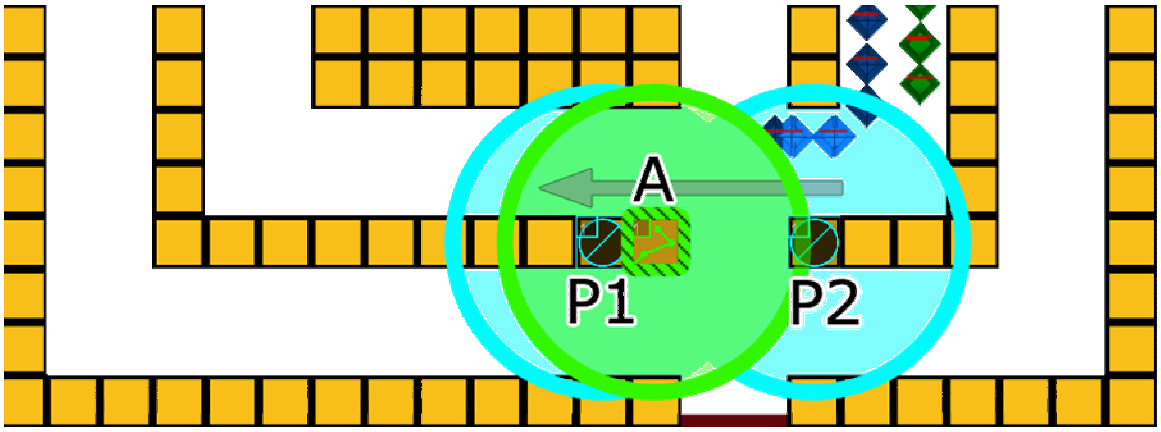


Figure 5.17. Goal-Directed Meaning of "Near". Position P1 is nearer placement anchor A than position P2 but when a subject says they want to place a slowing tower near the anchor, they mean position P2. They do not consider position P1 because it is after the offense tower at position A and therefore is of little value.

The statement "place a slowing tower near the offense tower" is shorthand for "place the slowing tower so that it makes the creeps slow while they are in range of the offense tower so that the offense tower has more time to attack them." The term "near" is in part a temporal term, referring to the two seconds that creeps will move slowly after being affected by the slowing tower and the time it takes them to reach the offense tower (note that "near" is also still a spatial term; terms can have multiple meanings concurrently; conflation of time and space is discussed in Section 6.1 Relationship of Space and Time). The creeps must pass position P2 before entering the range of the offense tower at placement anchor A and will reach that tower before the slowing effects from the slowing tower at P2 wear off, so the two are close and, more importantly, synergistic. In contrast, position PI is physically nearer to placement anchor A but, as defined by the creeps' direction of movement, is "behind" the position, meaning that the slowing tower will slow the creeps such that they are moving slowly after they can be attacked by the offense tower, making the effect, and therefore the position, useless.

5.5.2 Types of Representations

We define spatial features as visible, concrete properties of a spatial scene whose definitions depend on nothing more than their own spatial geometry. In this domain, spatial features include walls, path tiles, corners, U-turns and islands. We also include concrete spatial patterns. For example, subject E0100 always looked for a wall segment that was 4x2 and circled by both paths. The concrete spatial features we observed were always rotation-invariant but not size-invariant or path-direction invariant.

Spatial affordances are spatial properties that suggest or allow an action. Classic examples from the military domain include bottleneck, choke point, flank, core, front and rear, which can be partially determined from geometry but require other data such as direction relative to a second point. Examples that are partially defined by spatial information but strongly defined by function are areas designated potential kill zones, staging areas and rallying points.

Some spatial affordances are spatial features that have been identified as being "usable as" by another object. For example, Figure 5.15 highlights four spatial feature U-turns. Of those, only one (#3) can be used as a U-turn by a level 10 Green3 tower, while three (1-3) are U-turns from the perspective of a level 10 Red3 tower. For an agent who uses the strategy "place the tower on a U-turn", the number of U-turns on the map is dependent on the tower they're trying to place at that point in time. If the agent uses the same strategy twice with two different towers, the map will afford a different number of U-turns each time.

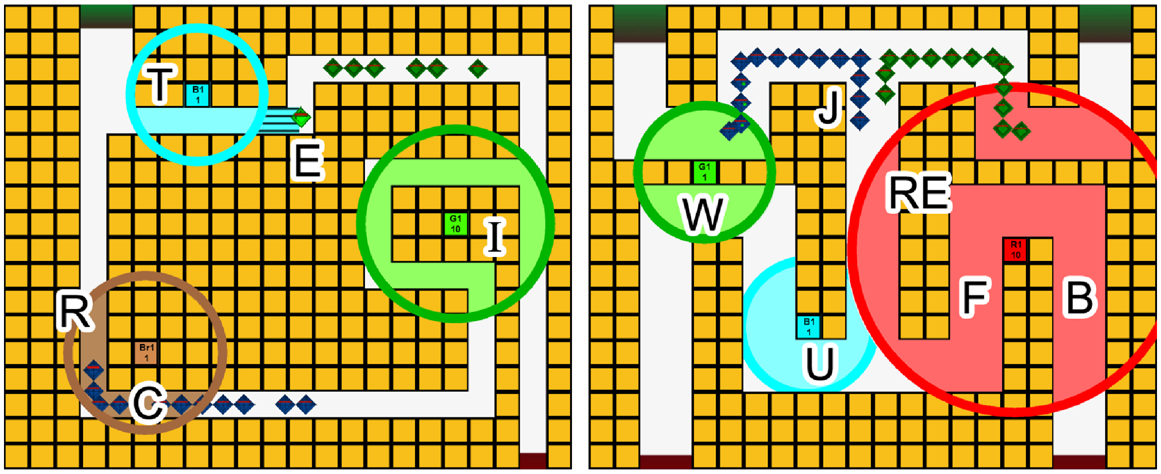


Figure 5.18. Types of Representations. B=Back. The back half of the R3-10 tower's range. E=Effect. A creep slowed by the tower at position T remains slow for the amount of time it takes to reach the position of the creep at E. F=Front. The front half of the R3-10 tower's range. I=Island. The area is, to the G3-10 tower placed there, an island because the tower's range reaches all four sides. J=Path Join. The two lines of creeps start at opposite corners and come together at position J. R=Range. Typically the agent is referring to the tower's usable range, which is the part of the range that covers path tiles. RE=Range Entry. This is the spot where the creeps first enter the range of the R3-10 tower's second attack window. T=Tower. U=U-Turn. This area is a U-turn for the tower at position U. W=Attack Window. The tower at position W has two attack windows, two disconnected path areas it covers.

Spatial affordances are not limited to usable spatial features. Most of the strategies we observed relied upon properties of objects. The most common property used was the range of a tower (see Figure 5.18, R). A tower's range is sometimes sub-divided into separate areas called attack windows (Figure 5.18, W). These properties can, in turn, have their own property. For example, Figure 5.18, RE, shows the secondary property `Tower.AttackWindow[2].RangeEntrance`, the point where a line of creeps first enters into the second attack window of the tower.

Some spatial affordances are based on meta-data. Examples include places where the two lines of creeps split and head in separate directions and places where the two paths join together (Figure 5.18, J). It is not clear whether these should be classified as spatial or not. They do, however, afford spatial actions since they identify a point where a tower (range, attack window, range entry point, etc.) can be placed.

A few affordances are based on status effects. For example, in Figure 5.18, E, the shaded region indicates locations on the map where creeps will be moving at half speed. It is more technically accurate to phrase this as a temporal quantity and say that the slowing tower at position T slows creeps for two seconds but because the creep follows the path from the tower to the right, the temporal effect can be mapped to a spatial location. Subjects sometimes referred to this as the "slowed zone". This is one of many examples in which subjects used spatial and temporal quantities interchangeably.

Many affordances are ordinal properties. The most common is usable range – how many path tiles are inside the tower's range? This specific example can be considered a quantification of concrete spatial data but not all ordinal properties are. Section 6.2.1 Usable Range discusses several ways of measuring range, some of which are not spatial. Another common quantity subjects used was path synchronization gap, which quantifies how far one line of creeps is behind the other. Although it is often measured in number of tiles, this quantity is important for a number of temporal strategies.

Although not as commonly used, several subjects based strategies on affordances generated by holistic analysis. This most commonly meant coverage. A typical strategy of this type would be phrased as "make sure the creeps are always being attacked". The goal is to make sure that the creeps are never at a place where a tower can't attack them but the action is to place towers so that they cover the entire path and the spatial affordance of interest is a path area that is not in range of any attack tower.

5.6 Problem Solving

It had been argued in educational psychology that the ability to form abstractions is the key to human intelligence. Significantly less attention has been paid to the other part of the equation – concrete problems cannot be solved with abstract solutions, the abstractions must be converted back into concrete actions and information. It has been argued that the concretization process is just as difficult as the abstraction problem and might be a significant factor in why, as one example, new engineers armed with a significant amount of abstract theory and knowledge but little practice with instantiating those abstractions have a hard time solving real-world problems their first years out of school.

Humans are able to solve a wide array of novel problems with minimal experience. We have argued that, at least in this domain but likely many others, this ability is due to the use of sets of abstract strategies paired to problem representations based on abstract concepts such as meta-data and affordances. The focus of the computational work presented here was on solving the abstract-to-concrete problem. We developed a set of agents that could take a problem, a set of tools (in this case, towers) and a set of strategies and generate a solution. A successful agent would successfully instantiate the given strategies on a wide variety of problems, including problems that are structurally quite different than any the agent had seen before.

Figure 5.19 shows the intended system design. It differs from the implemented model in that it shows the strategy and tower selection strategies that were not implemented for the reasons given above. It differs from the human model in that it is not iterative. Humans placed a subset of their towers, mentally simulated what might happen and then, if they were unsatisfied, moved the towers and occasionally changed strategies. For example, a subject might decide they wish to place a tower where it can maximize the number of times the creeps pass it (see STRATEGY SP5: MAXIMUM NUMBER OF PASSES). This decision is based on the belief that a particular location on the map affords a higher number of passes than other locations. Upon placing the tower, the subject studies the tower's range area to determine just how much additional time the tower has obtained. If the amount is determined to be insignificant, the subject will abandon the Maximum Number of Passes strategy and try a different one. Humans do this because they are unable to accurately forecast the results of their actions. This iterative process was not implemented in the computer model because the computer knows with certainty items humans needed to measure and mentally simulate such as the amount of path area covered, the amount of time the creeps will stay in range, etc.

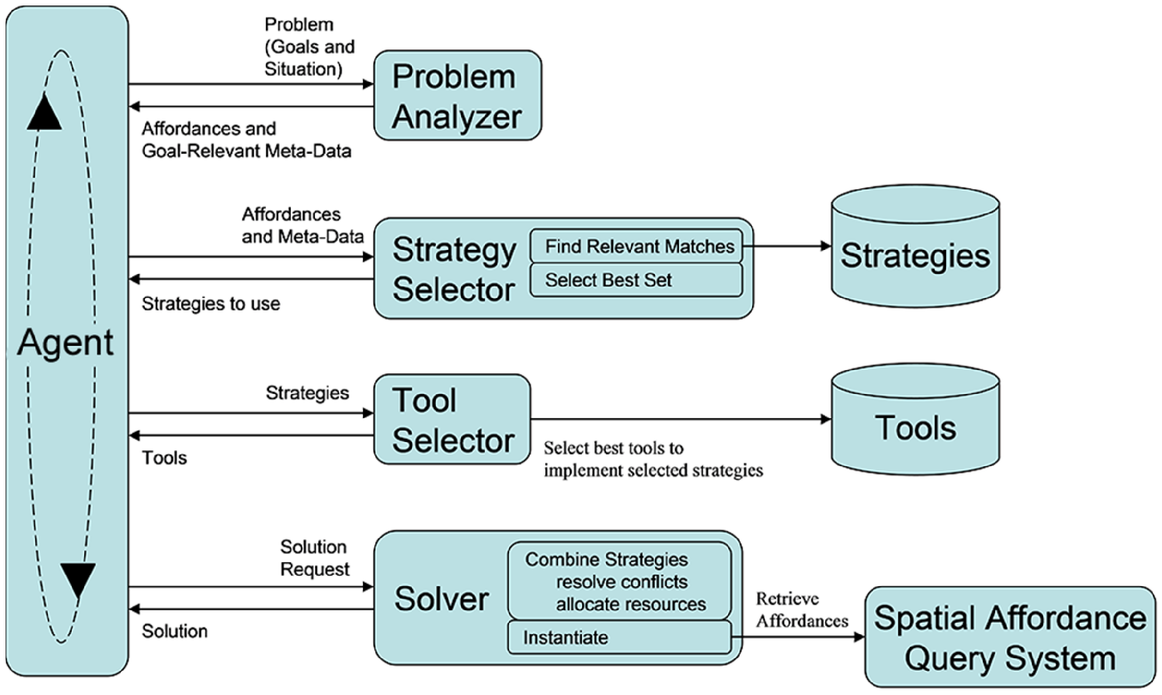

Agent
Problem (Goals and Situation)
Affordances and Goal-Relevant Meta-Data
Problem Analyzer
Affordances and Meta-Data
Strategies to use
Strategy Selector
Find Relevant Matches
Select Best Set
Strategies
Strategies
Tools
Tool Selector
Select best tools to implement selected strategies
Tools
Solution Request
Solution
Solver
Combine Strategies resolve conflicts allocate resources
Instantiate
Retrieve Affordances
Spatial Affordance Query System

Figure 5.19. Proposed Problem Solving Process. To solve problems without assistance, we believe an agent must be able to leverage its knowledge of tools and strategies to determine the affordances in a problem, use these affordances to select the proper strategies and tools and then combine and instantiate those strategies to create a solution. Not all of these capabilities were implemented in the current system.

The system depicted in Figure 5.19 breaks problem solving into four major steps.

5.6.1 Problem Analysis

The agent is given a problem. In this domain, the agent begins with the knowledge of the goal (stop the creeps) and actions (place towers). For each problem, the agent is given raw map information – a two dimensional array representing the location of wall tiles and a list of waypoint nodes representing each path. This information is given to the Problem Analyzer,which converts it into a format better suited for reasoning. In this domain this means marking each tile with information such as whether it is on a path, is adjacent to a path, is a corner, is where two paths merge, is where two paths split, etc. Later, as the solution is created, the annotated map stores the tower locations and information derived from that such as, for each tile, which offense towers cover it (i.e., which towers it is inside the range of), which slowing towers cover it, whether it is the entry point to a given tower's range, etc. The annotated map supports queries on this data – return all unoccupied wall tiles, corners, path tiles more than 75% of the way through the map, wall tiles where a specified tower can cover a path, unoccupied wall tiles where a specified tower can cover the last 25% of path 1, etc.

When a subject was given a new problem, almost all began by quickly taking a holistic view of the map. In some cases, the subject explicitly stated that they were trying to imagine the creeps moving through it. It is unknown whether they did this mental simulation for all maps, but most subjects at least verbalized doing so on the map The Void, a map that is unusual in that the two paths take different routes and are of significantly different lengths (this map also caused many subjects to develop the concept of path synchronization gap; all but one subject who recognized this concept immediately came up with STRATEGY TP8: EXPLOIT GEOMETRY and used it on this and subsequent maps). The initial holistic examination appeared to be looking for meaningful novelty – if a map had a highly unusual configuration, subjects tried to determine if it impacted them in any way and decide (as in the case of The Void) if it afforded new strategies.

After the initial holistic examination, how subjects studied the maps depended upon their preferred strategies. If subjects used strategies that relied on local properties such as useable range, number of passes or the existence of corners, U-turns, etc., they did not need to study the map holistically. Other strategies, especially temporal ones, required further analysis of holistic properties such as path length, path adjacency (i.e., splitting, separation and joining) and the location and stability of path synchronization gaps. The majority of subjects, regardless of preferred strategies, at least noticed "fast maps", that is, maps where the paths were unusually short, and reacted by placing slowing towers before they placed offense towers (no subject did this on any non-fast map). It is not clear, however, that this shows additional gestalt processing; in addition to being fast maps, these maps (2 in 1 Out, Bottleneck) lacked many of the affordances present in other maps, making the attribution for this change in piece placement order unclear.

The important part of this process is that, after the initial holistic analysis, subjects appeared to only look for those properties that were relevant for their strategies. For example, subjects who used the strategy USE CORNERS paid close attention to the number, position and type (two-sided, interior, etc.) of corners and did not worry about irrelevant properties such as path length. Those who used STRATEGY TP2: SWAMP focused on path length (the swamp strategy attempts to separate and freeze creeps and requires paths of a certain length to catch all of them) and did not pay undue attention to corners. When analyzing the map, subjects appeared to focus exclusively on those spatial or spatio-temporal features that were needed by the strategies they typically used. Our belief is that this reflects two important concepts. First, the goal of problem analysis is to re-represent the problem solely in terms of features that allow subjects to use strategies that help them achieve their goals. In other words, the outcome of problem analysis is a set of affordances. Second, problem analysis is a form of satisficing. The subject does not study the map for properties that have no potential value to them. Although human subjects could analyze or represent other properties, these properties would not lead to different behavior (i.e., it would not make them choose different strategies if they did not know or value any strategies that used those properties) and would make the processing task more difficult, both in terms of processing time and demands on their working memory. Because of how fast these analyses can be performed (each is a search for a specific and often simple spatial or spatio-temporal property) and the general processing power and memory of modern computers, the resource limitations argument for satisficing is not important for computer agents. While it is possible that this would not be true for problems in other domains (although our initial belief is that, for any domain in which humans do well naturally, it is), we did not attempt to model this particular satisficing process.

5.6.2 Strategy Selector

The goal of the problem analysis process is to re-represent problems in a form useful to strategies. For example, STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION places a tower where its range reaches two non-contiguous areas (i.e., attack windows) and the time to move from the first area to the second (i.e., temporal separation) is greater than a specified amount of time. The problem analysis process returns a problem representation that can quickly answer queries such as "are there any positions that provide the required temporal separation?" Given this representation, the Strategy Selector selects the set of strategies to use to solve the problem.

We can only speculate about how this process works. We did not create a computer model of this process for several reasons. First, subjects were unexpectedly consistent with their choice of strategies. Only two subjects selected different strategies and towers for different problems, giving us little data to work with. Our belief is that this is a training issue. The majority of subjects spent roughly an hour in training, their time spent learning how the puzzle works, how the pieces worked, what strategies were possible, etc. Of the two subjects who used different strategies for different problems, one (the author) had significant previous experience with the domain. If all subjects had had significantly more experience with the domain, perhaps we would have seen significantly more context-dependent strategy selection.

It is worth noting that all subjects switched strategies when their preferred strategies could not be instantiated. For example, if a subject's strategy was to USE U-TURNS and the map contained none, the subject was forced to try something different. The agents we created lacked a sophisticated method for switching strategies. If the agent could not apply their strategy, they fell back to a strategy (STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME) that was known to work on all problems.

While we did not computationally model the strategy selection process, we did model the representation of these strategies. A `PlacementStrategy` object specifies a criterion that candidate positions must match. They are implemented as filters. To place a tower on a corner where the usable range is greater than 20, you first pass the set of candidate positions to a strategy that finds corners and then pass the results of that to a strategy that returns the subset of those positions that match the range criterion. In this case, the order the strategies are executed in is unimportant, although it can be controlled.

`GroupStrategies` objects represent a set of placement strategies for a group of towers. A set of these objects is stored in a `SolutionRequest`. Consider the situation where the problem solver wishes to place their offense towers near the start of the map and where they cover the most usable range. To keep the creeps slow while they're being attacked, a set of slowing towers is placed around the offense towers along the path. There are two groups of towers, each with different placement requirements. The first group of towers to be placed is the offense towers. Placing them requires two placement strategies, one that selects those tiles that afford the most usable range and another to select tiles near the map entrance. All of this is stored in a group strategies object. To make sure these towers are placed first, they are the first group strategy object placed in the solution requests strategy queue. To place the slowing towers, a second `GroupStrategies` is created and added to the solution request strategy queue. This object would record the towers to be placed (the slowing towers) and the placement strategies that select tiles near the first group and along the path.

5.6.3 Tools Selector

In tower defense, the goal is to stop the creeps and the tools to do so are the towers. The third step of the problem solving process is selecting the set of towers to use. In Figure 5.19 we show this as being dependent on the strategies the problem solver has decided to use. In reality, the process is likely more collaborative and iterative. Certain strategies are only achievable by certain towers, meaning the towers offer affordances, both to the problem analyzer and the strategies.

We did not computationally model the tool selection strategies. One reason for this is that the selection of towers is an important but, to a degree, non-spatial process. In the human data we saw that tower selection makes a big impact on the problem solver's score. We removed tower selection from Experiment 2 so we could focus on spatial reasoning. Tool selection strategies and the way tower affordances interact with problem analysis and strategy selection is an interesting topic but one out of scope for the research described here.

A second reason for our not implementing this is that, while we captured a lot of human data on tower selection, it is mostly retrospective. Subjects had largely settled on a set of towers to use by the end of the training phase and we did not collect think aloud data during training. In testing subjects explained why they used the towers they did but we did not collect any significant information on how they came to form those preferences.

5.6.4 Solver

Once the agent has represented the problem in a manner that makes selecting and executing strategies easier and has chosen a set of tools and strategies, they send this information to the Solver. Figure 5.19 shows two parts to the solver.

Combine Strategies. First, the strategies must be combined. There are many ways to do this. In the computer model, strategies are executed in a specified order. If a tower should be near the exit and where the usable range is highest, one strategy would be the primary strategy and the second would be used as a tie breaker.

In some situations, however, strategies are mutually exclusive. One example is when two strategies are of equal priority and select different positions. For example, subject N0100 wanted positions that were a compromise of usable range and the number of times a creep moved through the tower's range. To handle this, a compromise must be formed. In our computer model, we addressed this by adding a new strategy, the meta-strategy `PlaceTowerOnCompromiseStrategy`. This strategy takes a set of strategies that return a list of scored positions and weights for each to create a composite score. Since scores are numeric, the weight was partially used to normalize units.

It is possible that there are strategies that cannot be combined with prioritization or compromise. We did not encounter any in our human data but this does not mean it can't happen, either here or in other domains. The solution is likely to reject one of the strategies. It is not clear whether that level of reconciliation should be handled in the Solver or the Strategy Selector. As we didn't implement either, we leave this issue for future work.

A second issue strategy combination must deal with is resource allocation. Every tower allocated to one strategy is one that is potentially not available to the other. As with compromise weighting, we saw many subjects deal with this problem and saw many different results. From the data we collected it is not clear how one decides to, say, allocate six slowing towers to a DIFFERENTIAL SLOWING region rather than five. It is also not clear that subjects had an explicit process for this – it appears that a lot of problem solving was placing towers, checking the result for a gut feeling and adjusting. While think aloud and interview data allowed us to understand many of those decisions, it was difficult to get detailed, quantifiable information about allocation and weighting.

For the implemented system, the combine strategies process was not needed as the strategies and tower allocations for each subject used were pre-selected. The chosen strategies were reconciled using strategy prioritization and compromise meta-strategies.

Instantiate. The core of the computational work in this research focused on strategy instantiation. Instantiation is the process of converting abstract strategies and concepts into a concrete solution. At a high level, the instantiation process iterates through each tower functional group (`GroupStrategies`) and, for each group, iterates through each placement strategy. This set of placement strategies merge to create a more complex strategy such as "place this tower at the position where its range entrance intersects the tile where the path synchronization gap is highest and the usable range is greater than 12 and the coverage balance is no worse than 0.75" (coverage balance is the proportion of time one path spends in range relative to the other). Each of the component strategies is of the form "find all positions that match criteria X", where criteria X is a spatial or spatio-temporal affordance. The bulk of the work (and intelligence) in these strategies is being able to extract the necessary information from the problem. This is done by submitting queries to the Spatial Affordance Query System. The job of this system is to use the lower-level affordance and meta-data information in the problem re-representation created by the problem analyzer (`MapAnalyzer`) to answer the affordance queries made by the solver.

Chapter 6

Unifying Spatial and Temporal Reasoning

Given a set of towers that do different amounts of damage, the goal of the tower defense task can be approximated by the temporal goal of maximizing the weighted sum of tower active times, with the weight being proportional to the tower's damage per second. Given that actions are spatial, the tower defense task can be thought of as using space to maximize time. The analysis in this thesis will proceed from this belief, although it must be noted that the temporal goal of maximizing active time is not perfectly correlated to the actual goal of creeps destroyed. See sections 8.4.2 To Convert Wounds to Kills and 8.4.3 To Reduce Overage for information on how damage distribution affects scores.

6.1 Relationship of Space and Time

6.1.1 Space-Time Conflation

In the tower defense task, the problem solver must use spatial actions to maximize a temporal quantity. How can space be used to manipulate time? For many subjects, the answer was to act as if they were the same thing.

Even though the ultimate goal was to maximize time, the majority of subjects talked about maximizing space. When describing their goals, they used language such as:

- "I want to cover as many… ***[path] squares*** as possible."
- "I choose corners because they overlook the most ***[path] area***."

A common strategy was to use slowing towers to slow the creeps and thus give the towers more time to fire. Although this clearly has a temporal effect, the language used was normally spatial, focusing more on keeping the creeps in a particular space rather than giving the towers additional time:

- "These two [slowing towers are here] mostly just to keep them ***in the range*** of the [offense towers] a little bit longer."
- "[U]se the [slowing towers] and put them near all the [offense towers] to slow them down… That way they're ***in range*** of the [offense towers] for as long as possible."

Some of the subjects noticed that the range of their tower was so large that, if the tower were placed next to the path, the end of the range area went past the path and covered a significant amount of wall space. The obvious solution was to back the tower away from the path. The explanation for this was almost always spatial:

- "I didn't put them on the end here because then the ***[range] circle*** here would go off into the wall so that range is being wasted so you get a little bit more on this end if you pull it back a square…"

Some subjects used strategies that we classify as temporal strategies. These include strategies whose goal is to split the two lines of creeps so that they go through a tower's range one at a time, giving the tower additional time to fire. Even these strategies were often described with spatial terminology:

- "I want to place my towers ***where the creeps are separated***."
- "I'm going to pull [slow] this line so that they ***fall behind***."

This was not always true. Some subjects explicitly talked about time. More interestingly, a few (although not many) implicitly recognized a relationship between space and time. In the following quote, the subject describes the targeting strategy T1: FOCUSED FIRE, in which the subject attempts to destroy individual creeps as fast as possible by having all of the towers focus on the same creep:

- "I'm trying to kill the creeps ***as fast as possible*** [temporal goal] and because the towers select the creep that is closest to them, I try to put the towers as near each other as possible [spatial action] so that they select the same creep."

While some subjects mentioned time, they rarely focused on it. Tower defense puzzles are temporal problems but subjects treated them as if they were spatial. Rather than maximizing time, most subjects chose to maximize space. Subjects appeared far more comfortable talking about space than time.

- "[P]lace all of the green ones where they had the most ***area*** covering where the creeps are walking."
- "The main thing I was worried about with the first map was where I wanted to get the most possible ***room***."
- "I thought, "How can I get the majority of my ***circle*** placed within the [path] corridor so that it maximized the amount of space that it could shoot at the barbarians as they were going through?", so for me this was all about trying to maximize the amount of the circle that was placed inside the corridor."

We are not the first to note that people use space as a proxy for time (Boroditsky 2000). We do, however, believe this is the first time that space-time conflation has been shown in a domain involving complex problem solving and is the first research to show how this conflation affects task performance.

6.1.2 The Case for Space-Time Conflation

Space and time are clearly distinct and independent concepts. By converting a temporal problem into a spatial one, subjects are solving an entirely different type of problem. How can such an approach be expected to be successful?

We divided placement strategies into spatial and temporal. 11.2 Effects of Spatio-Temporal Independence shows how these compare to a set of baseline strategies. Treating the temporal problem as a spatial one does remarkably well, significantly outperforming the random baseline. Why?

The answer is that there is a linear relationship between the length of a path and the amount of time it takes a creep to traverse it. Once again, consider the simple case where there is one tower and one creep. The tower fires once per second and does five points of damage per shot. The creep moves at one tile per second. If the tower is placed at a location where the path through its range is five tiles long, the creep will take five seconds to move through the range, the tower will be active for five seconds and the tower will inflict 25 points of damage on the creep. If the tower is placed at a location where the path through its range is twice as long, it will take the creep twice as long to move through it and the creep will take twice as much damage. The number of creeps destroyed is proportional to the amount of time the tower fires which is proportional to the amount of path area it covers. As the tower covers more room, it is active for more time, fires more times, does more damage and stops more creeps.

The purely spatial AI MAXIMUM USABLE RANGE strategy scored 91% higher than the random baseline agent. Clearly, re-representing temporal problems as spatial problems is highly beneficial.

6.1.3 The Case Against Space-Time Conflation

Given that humans seem more comfortable reasoning about space than time and space appears to correlate with time, is there a reason to explicitly focus on time?

To answer this, we looked at the performance of temporal strategies. The results are presented in Chapter 11 Results of Experiment 2 but the short answer is, yes. Subjects who treated space and time as independent often sacrificed space in an attempt to increase time. These strategies outperformed the spatial strategies by 23%. Why? Because contrary to what many subjects believed and in spite of the success of the spatial strategies, space and time are neither correlated nor proportional. If there is only one creep, path length and tower activation time are correlated. When there are multiple creeps, the situation gets more complicated.

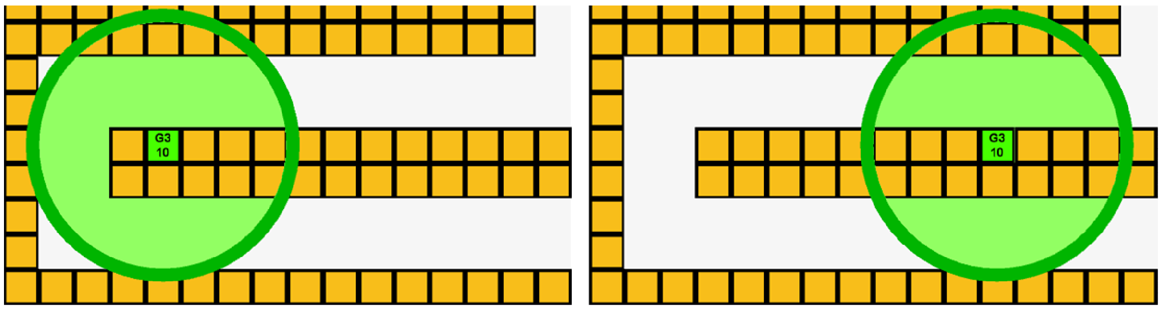


Figure 6.1. Space vs. Time. The tower on the left covers more space. The tower on the right, however, is active for a longer period of time.

The remainder of this chapter deals with the question of space vs. time and the unique issues that multi-agent movement through space raise for temporal reasoning.

6.2 Temporal Extensions to Spatial Representations

Chapter 7 Solution Representation lists all of the representation elements we observed in our testing and Chapter 8 Strategies discusses how they were used. Most (although not all) of these focus on increasing space or time. At their core, they are trying to increase usable range or attack window size. In this section we discuss these two fundamental concepts in more depth.

6.2.1 Usable Range

The goal in tower defense is to get the highest score possible. Because the score is based on the number of creeps that make it through the map, the related goal is to destroy as many creeps as possible. The problem solver has no control over the amount of damage a tower does (each tower does a fixed amount per shot), how quickly the tower fires (each tower has a fixed fire rate) or what causes the tower to fire (the tower always fires when a creep is in range, never fires when it is not). The number of points of damage a creep must take to be stopped is fixed as is the creep behavior (i.e., it follows the path; creeps do not react to the environment, neither dodging towers nor retreating). The one variable that the problem solver can influence is the tower's active time - the more time a creep is in range, the more often a tower fires. A tower placed where creeps spend two seconds in range will fire twice as often as a tower placed where creeps are only in range for one second. The only direct action a problem solver can take is to place a tower on the map, a spatial action, but the goal to be maximized is active time, which is a temporal property. (note that active time is not perfectly correlated with score as it ignores the issue of damage distribution and overage; see Sections 8.4.2 To Convert Wounds to Kills and 8.4.3 To Reduce Overage for more information).

An observation made by most subjects was that the larger the usable range (the part of the range area that covered the path), the longer it took the creeps to move through the range and therefore the more time the tower has to attack. Based on this, many subjects tried to maximize the towers' usable range.

This is not to say that they succeeded in doing so. We observed several reasons for subjects picking positions that led to large, but not maximal, usable range, including satisficing and inability to make precise measurements. An unexpected reason was that *usable range* turned out to be a surprisingly difficult concept to define.

Each type of tower has a maximum distance it can fire. This is measured in world units, which are significantly smaller than the grid tiles that towers are placed on, resulting in towers covering portions of tiles. A creep is in range only when its center point is in range. If a small part of a tile is in range but the center of the creep is not, that tile is not considered to be in range. Because it does not significantly alter the analysis, when reporting the number of tiles in range we round to the nearest half tile.

There are two important things to understand about a tower's range.

First, although some subjects took it into account, the total number of tiles in a tower's range is not an important measure. Towers only attack when a creep is in range and creeps only move across tiles that are on their path. Non-path tiles (i.e., wall tiles and empty tiles) do not in any way contribute to a tower's active time, number of shots fired or amount of damage done.

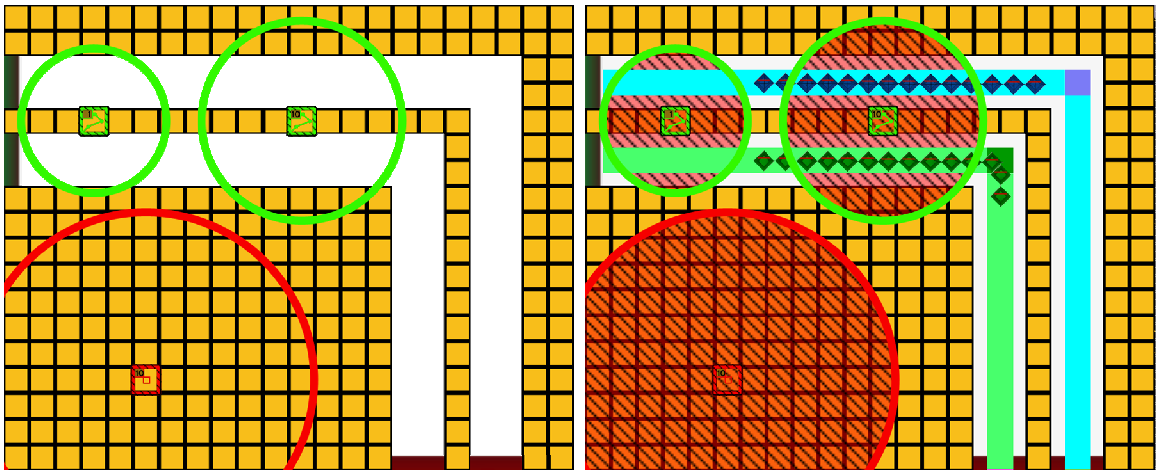


Figure 6.2. Range vs. Usable Range. LEFT: Circles surround the tiles that are in range of each tower (a G3-1, G3-10 and R3-10 respectively). RIGHT: Creeps only traverse tiles that are along their paths (blue and green lines). Red shaded areas are in range but will never be traversed by a creep, making them of no value to the tower or the player.

	Range Radius	Tiles in Range	Path Tiles in Range
Green3-1	70 world units	13 tiles	10 tiles
Green3-10	97 world units	36 tiles	14 tiles
Red3-10	213 world units	143 tiles	0 tiles

Table 6.1. Usable Range of Figure Figure 6.2. The range of three towers (a level 1 Green3, level 10 Green3 and level 10 Red3) as the maximum distance from the tower in world units, the number of tiles in range and the number of path tiles in range for the towers in Figure 6.2.

Second, the number of path tiles in range does not determine how often a tower fires. This number is based on several things.

One is the number of creeps that move over a tile. While the two lines of creeps often move side-by-side, this is not always true. In Figure 6.3, left, the hallway is two tiles wide (vertically) but one line of creeps moves down the center of the area and the other line doesn't move down it at all. There are 13 physical, non-wall tiles in range but only seven path tiles. No more than seven tiles worth of creeps will be in range at any given time. In contrast, the initial area covered by the tower in Figure 6.3, right, is 46% smaller

(7 vs. 13 physical tiles) but, because both lines of creeps move over the same tiles, it can hold twice as many creeps (14 tiles worth vs. 7).

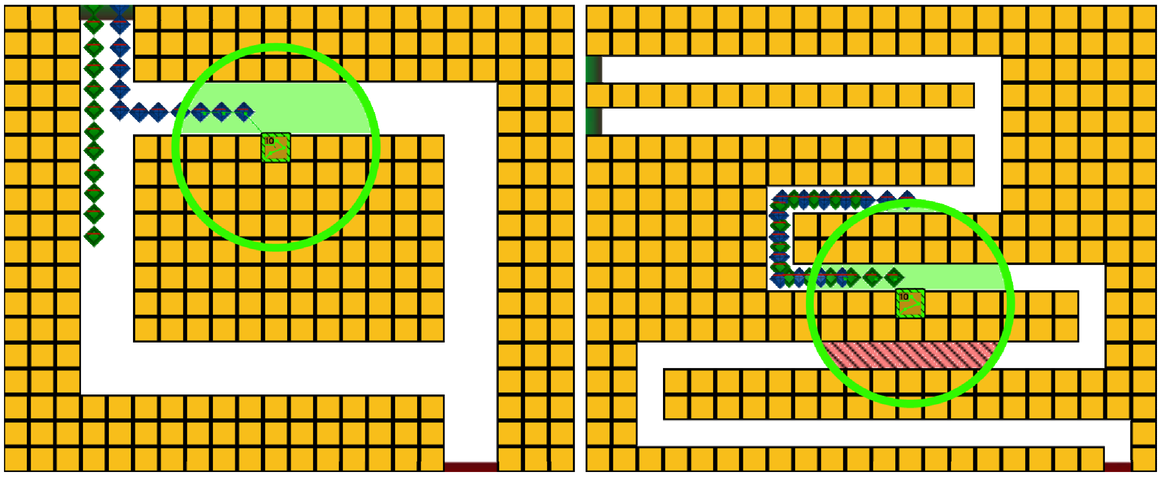


Figure 6.3. Path Volume. The tower on the left covers twice as many non-wall tiles than the tower on the right (not including the second attack window, the red, striped area) but is worth half as much. LEFT: The non-wall area covered by the tower is two tiles high but only one line of creeps moves through it. RIGHT: The non-wall area covered by the tower (green shaded area) is only one tile high but both lines of creeps move over it.

	Non-Wall Tiles in Range	Path Tiles in Range	Maximum Number of Creeps in Range
Left Tower	13	7	8.75
Right Tower	7	7	17.5

Table 6.2. Path Volume for Figure 6.3. The range of three towers (a level 1 Green3, level 10 Green3 and level 10 Red3) as the maximum distance from the tower in world units, the number of tiles in range and the number of path tiles in range for the towers in Figure 6.3.

Another factor is how often the creeps move over each tile. In the map Round the Twist (Figure 6.4), the path crosses over itself, causing creeps to move twice over the intersecting path tiles and giving the tower two chances to attack creeps on those tiles.

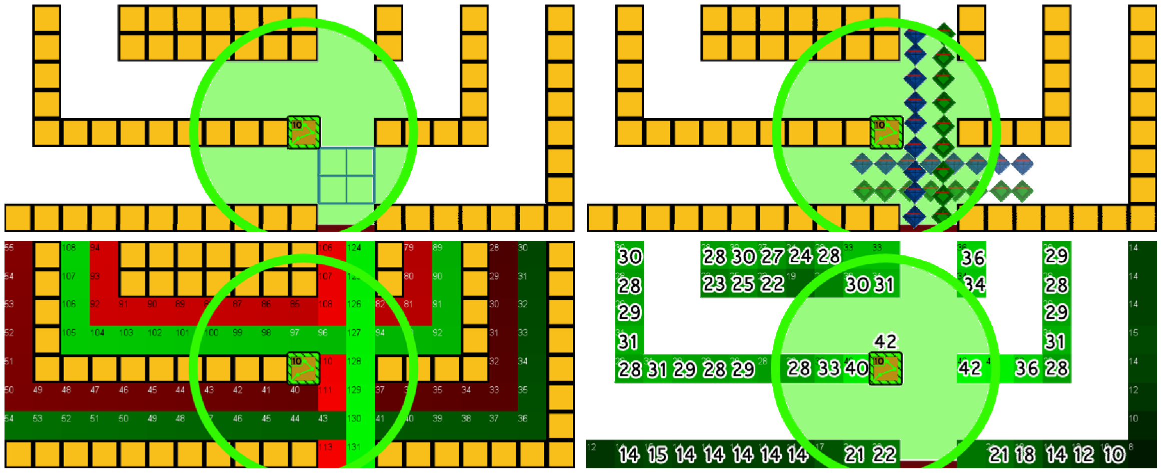


Figure 6.4. Different Views of a Path. TOP LEFT: The four tiles indicated by the blue squares are an intersection. TOP RIGHT: Creeps pass over the intersection tiles twice (image is retouched to show the two lines of creeps at different points in time). BOTTOM LEFT: Green and red path tiles show the step number (order they are traversed), indicating the flow of time over those tiles. The color gets brighter near the end of the path. The paths overlap at the intersection. BOTTOM RIGHT: Wall tiles show the usable range of towers placed at those positions. The position at the intersection offers significantly more usable range in part because the intersection tiles are counted twice.

The above constitutes the definition of usable range, which is based on a measure we call path volume. Rather than count the number of path or non-wall tiles in range, path volume counts the number of times a creep, from either line, passed over any in-range tile.

A third and very important factor related to the amount of time a tower fires is discussed in Section 6.2.2 Attack Windows.

6.2.2 Attack Windows

Usable range defines the area where a tower can attack creeps. This is a purely spatial property. It refers to the total amount of path area a tower covers, regardless of whether those areas are separated or continuous. Attack window refers to a continuous "opportunity to attack". A specific type of this is what subjects called a "pass". Consider the example of a wall where a tower's range covers both sides of the wall. Unless the tower is covering a U-turn, the tower will have two opportunities to attack – once when the creeps move along one side of the wall and later when the creeps move along the other side of the wall (Figure 6.5). A tower has as many attack windows as it has separate areas it covers.

An attack window is not necessarily a subdivision of a tower's usable range. Attack windows are defined either from the perspective of a single creep or a set of creeps. From the point of view of a single creep, an attack window is simply a subdivision of the tower's usable range. The combined size of the attack windows is linearly proportional to the size of the usable range and the number, size and location of the attack windows are defined without context. We will call this the *per-agent attack window*.

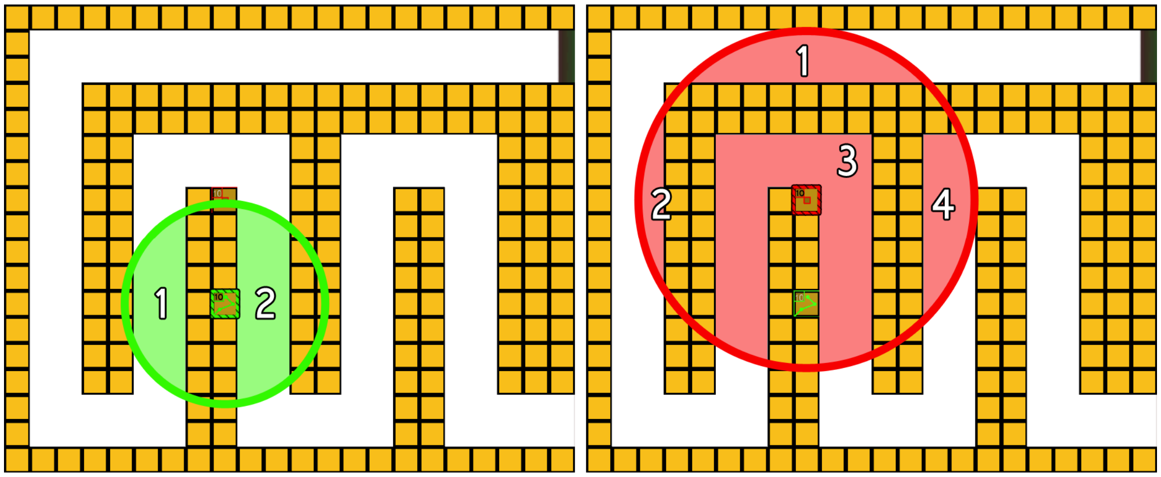


Figure 6.5. Attack Windows as Seen by a Single Agent.

From the perspective of a group of creeps, two areas are contiguous if the group can be in both at the same time (see Figure 6.6). The definition of a group attack window is context-dependent – the number, size and location of the attack windows can only be defined by knowing the size of the group. Which creeps constitute a group depends on the "distance" between attack windows – if two creeps are in two different per-agent attack windows at the same time, they are part of the same group and if a group is in both per-agent attack windows at the same time, they are a single *group attack window*. The size of a group attack window is based partially on the tower's usable range but also on the agent group size and distance between per-agent windows. Unlike the per-agent attack window, the size of a group attack window can be changed by moving the tower (independent of changes in usable range) and using geometry and slowing towers to stretch, compress or separate a group of creeps.

When subjects referred to attack windows, most appeared to be referring to per-agent attack windows. They thought about these primarily in spatial terms. This is perhaps not surprising given that the number, size and location of per-agent attack windows are easy to measure. Since they are discontinuous spatial areas, they are identified automatically by the visual processing system and require no explicit mental effort on the part of the subject.

In contrast, subjects who reasoned with group attack windows thought about them in temporal terms. They tried to determine whether a set of per-agent windows were temporally contiguous, meaning they formed a single time period during which the tower of interest was active. Measuring the number, size and location of group attack windows is non-trivial. Unlike per-agent measurements, group measurements do not automatically co-opt the visual processing system. Measuring temporal contiguity requires conscious effort on the part of the problem solver.

For those who used the concept of group attack window, two observations were common. First, subjects noted that if a group of creeps were stretched out (e.g., one line of creeps fell behind another for some reason), it took longer for the group to move through the tower's range, giving the tower more time to attack.

Second, because a tower can only attack a fixed number of targets at one time, they looked at whether a group of creeps would enter a tower's range while that tower was still busy attacking other creeps. If so, the creeps just entering the range would be able to move through the tower's range unharmed while the tower was busy with the other creeps.

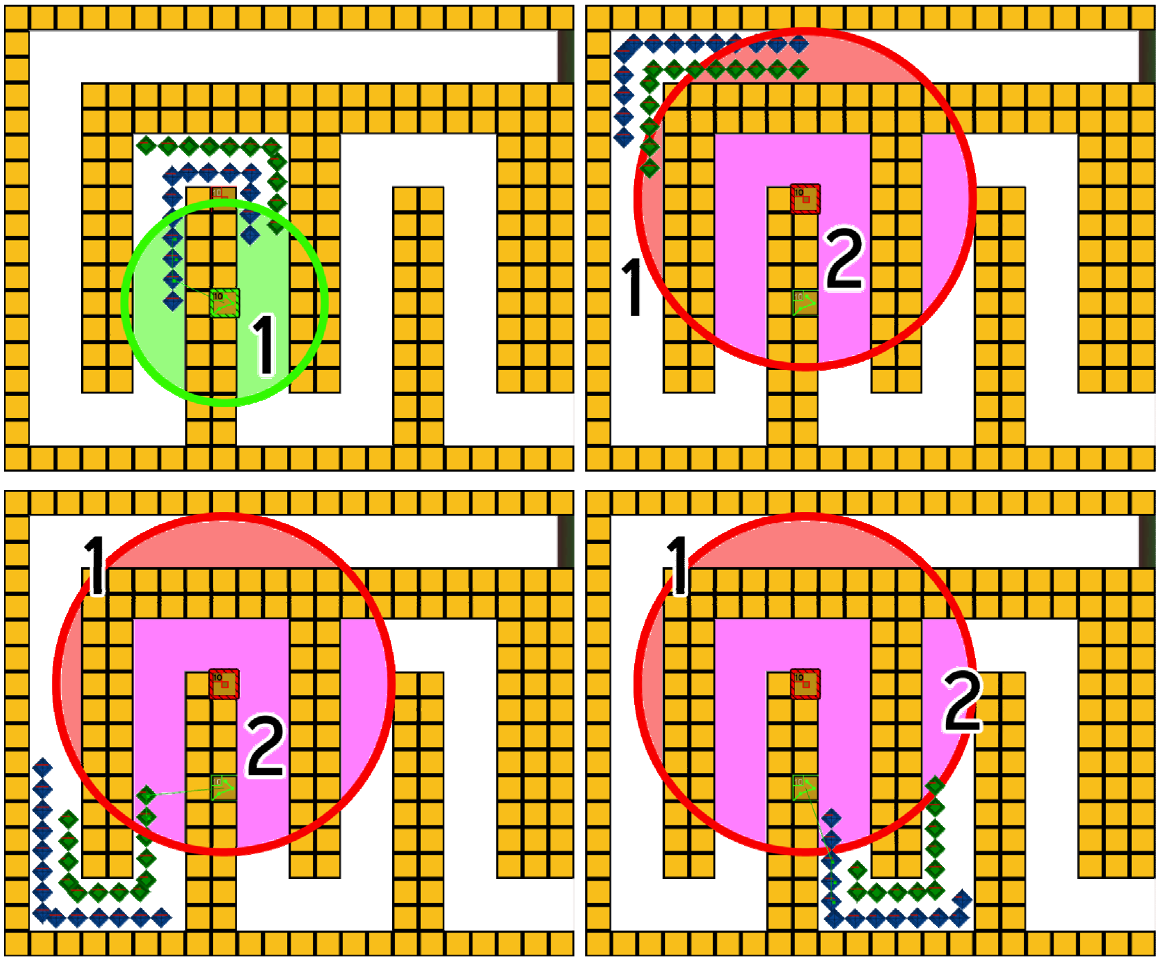


Figure 6.6. Temporal Group Attack Windows. TOP LEFT: There are agents in both spatial, per-creep attack windows at the same time. There is only one temporal, group attack window as the tower has one long, continuous period of time to attack rather than two smaller, disconnected time periods. TOP RIGHT: Agents are in the first and second spatial, per-creep attack windows (red) at the same time, forming a single temporal group attack window. BOTTOM LEFT: The group of agents completely exits the first two spatial windows before any of them enters the third window. The third spatial attack window is the start of a second temporal, group attack window. BOTTOM RIGHT: Agents are in the third and fourth spatial windows at the same time, ensuring that there is no gap in the tower's active time when creeps are in between the two spatial windows.

A third, though less common, observation is that towers needed "time to reset", meaning that there should be a break between times the tower attacks a group so that the effects of battle can be erased (although this might seem like an odd comment, a quantitative explanation for this is offered in Section 6.4.6 Buffering and Attack Window Decay).

A method for measuring group attack window size and performance implications is given in Section 6.3 A Unified Measure for Space and Time. Manipulations of group attack window size underlie the majority of the temporal placement strategies in 8.2 Temporal Placement Strategies.

6.3 A Unified Measure for Space and Time

6.3.1 The Need for a Unified Measure

There are many strategies that one can use. How does one know which is best? How does one decide that placing a tower on a U-turn is better than placing it on a normal wall? Some subjects justified the choice by saying the tower on a U-turn covered three sides while a normal, two-sided wall covers only two (see Figure 6.7). Although quantitative (i.e., number of sides covered), there are problems with this reasoning. First, the value is not directly tied to a useful value. A U-turn covers more sides of a wall than a normal wall position but that doesn't guarantee that it covers more path area. Second, it doesn't explain why corners are preferred to normal wall spots; subjects tended to prefer corners but both corners and normal wall spots cover two path segments. Third, while maximizing the number of segments covered allows one to choose between corners and U-turns, it provides no mechanism for choosing between two instances of a given feature (e.g., two U-turns). Fourth, the measure is purely spatial and does not take into account temporal phenomena (this is not as damaging as it might initially seem; see Section 11.2.1 Effectiveness of Spatial-Proxying).

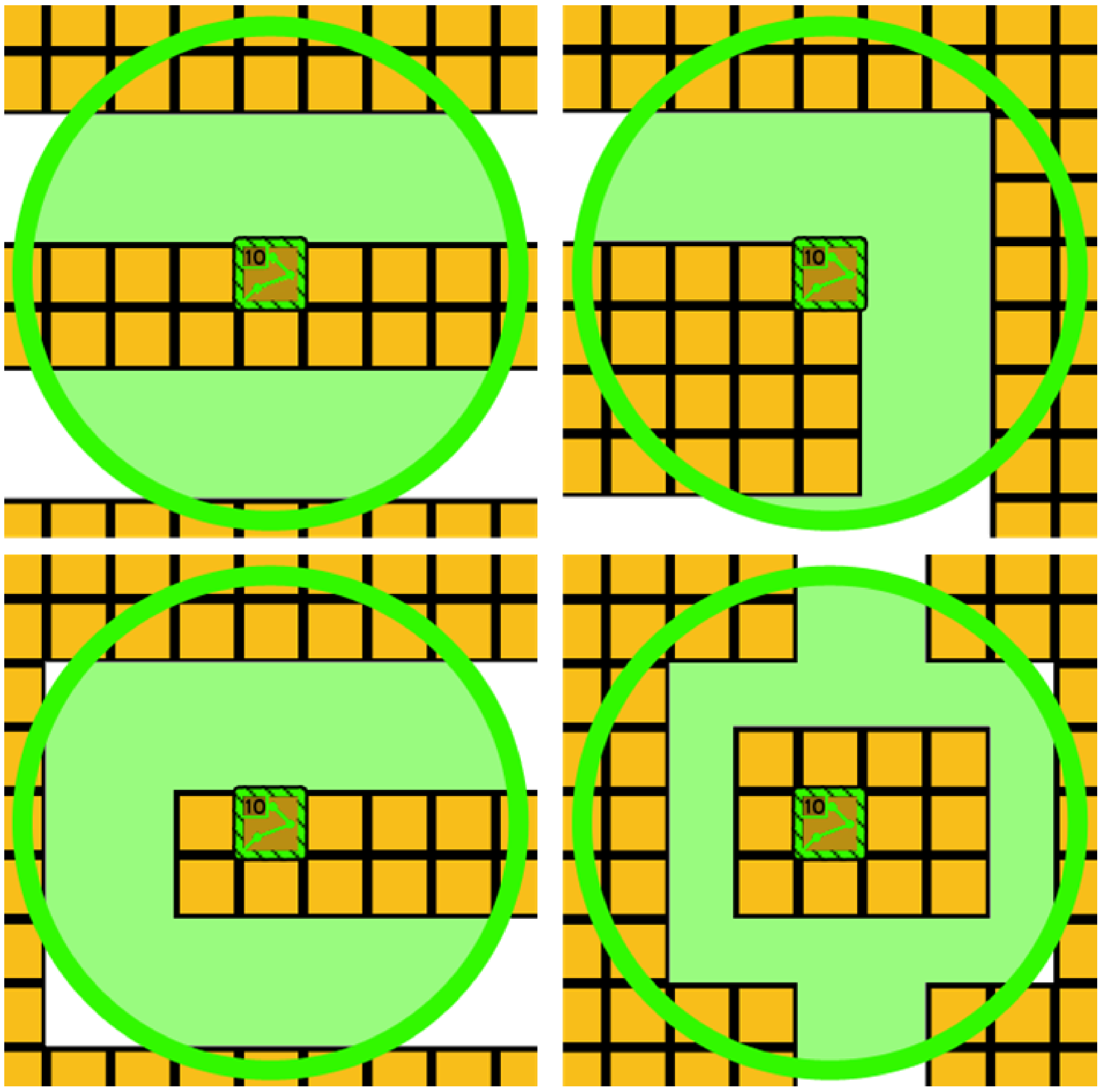


Figure 6.7. Four Different Spatial Affordances. TOP LEFT: Normal two-sided wall. TOP RIGHT: Single-sided corner. BOTTOM LEFT: U-turn. BOTTOM LEFT: Island.

Finally, this form of quantitative measure is imprecise to the point of essentially being qualitative. While this form of reasoning is easier for humans to do, there are two reasons why we want a more precise measure. First, if our aim is to build better computer agents, quantitative reasoning is easier than qualitative. Second, qualitative reasoning is not precise enough to explain why certain strategies work better than others. This might be acceptable if the predictions made by the qualitative model (more path segments or path area mean higher scores) were intuitive and correct, but they are not. In the next section we show how quantitative analysis predicts a correct but counter-intuitive result.

An alternative to judging a position by the number of path segments covered by a tower placed there is to estimate the amount of path area covered. This is usually done through visual estimation, with many subjects moving a tower from position to position and judging which looks the largest. Subjects were not particularly good at this. They often missed the highest value areas when those areas were in an unexpected spatial configuration (e.g., two sided corners, as shown in Figure 6.8).

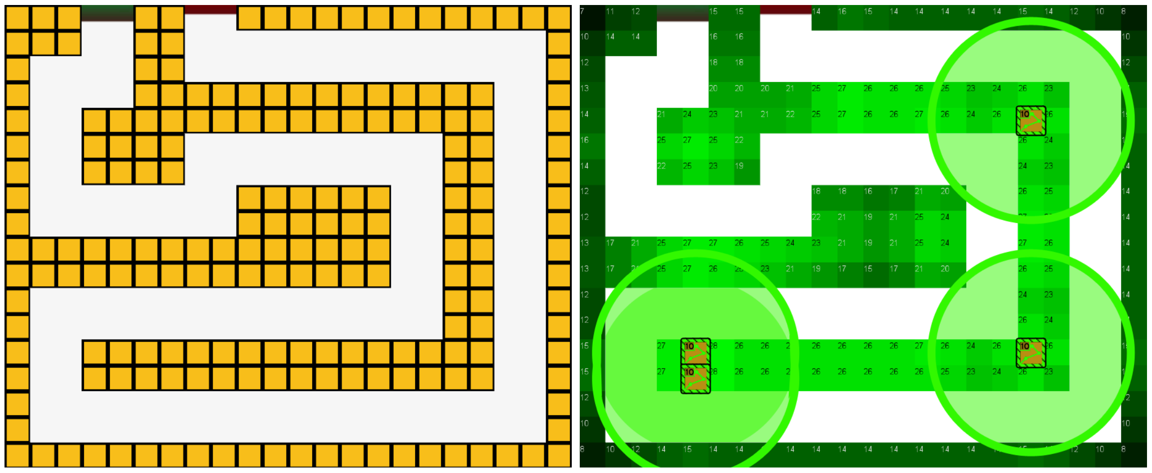


Figure 6.8. Maximum Usable Range is Hard for Humans to Measure. LEFT: The map Elemental-ish. RIGHT: Towers placed at the four positions where towers will cover the most path area. Subjects often select the U-turn but very few selected the highlighted corners.

Even if humans always measured usable range directly and were able to accurately measure it, it would not directly measure the value of interest because it measures only space, not time. The goal is to stop the creeps, which can only be done by damaging them with tower attacks. Because each attack made by a given type of tower does the same amount of damage, given a fixed set of towers, the variable to maximize is the number of shots the tower makes. Since the tower fires at a fixed rate of fire, the correlated goal is to maximize the number of times a tower fires. Since towers only fire when creeps are in range, the correlated goal is to maximize the number of time intervals in which a creep is in a tower's range (note that this is not entirely true; the tower's attacks must be allocated properly – a given amount of damage can either destroy one creep or wound many, and no points are earned for wounding a creep).

The common sense assumption made by many subjects is that if a tower covers more path area, it takes the creeps longer to move through the tower's range and thus the creeps spend more time in the tower's range, causing the tower to attack more often and therefore do more damage. Increasing space increases time increases the score.

While this belief was both popular and considered obvious, we wished to verify this quantitatively. This led to the development of the al measure.

6.3.2 The al Measure

The size of a tower's attack window can be measured spatially (e.g., the number of tiles a creep must move over) or temporally (e.g., the number of seconds it takes a creep to move through the area). For a single creep, assuming that it moves at a fixed rate of speed, the spatial and temporal measures are linearly correlated and spatial measurements can be directly converted to time and vice versa. For example, assuming that a creep moves one tile per second, an attack window five tiles wide (a spatial measure) takes five seconds to cross (a temporal measure) and vice versa.

This relationship does not hold for a group of creeps. Assuming that a creep moves one tile per second and there are 14 creeps in a line, an attack window five tiles wide does not take five seconds to cross, nor does it take 14*5=70 seconds to cross since the creeps do not move one at a time, they move concurrently. What we want is a measure of the size of an attack window relative to a group of creeps that can be converted to space or time and which allows mapping from space to time and vice versa.

We have chosen to use as the basis of measurement the amount of time it takes a single, unmodified creep to move its own length (creeps are not necessarily the same width as a tile; in GopherTD, a line of 14 creeps is 14 creeps long but only 11.2 tiles long; we use creep length as it is a more natural measure of the length of a line of creeps and slightly simplifies the math). We refer to this as the agent length, *al*. This measurement can be directly converted to space and time. For example, assuming a creep is half a tile wide and moves at half a tile per second, an attack window that is 5al wide is 2.5 tiles wide and takes 2.5 seconds for a single creep to cross.

While simple, the al measure allows us to directly measure the amount of time a set of creeps spends in a tower's range. Purely spatial measurements such as path length can tell us the temporal size of an attack window for a single creep but not for a set. The value of measuring attack windows in terms of the time it takes a creep to move its own body length becomes clear in the next section.

6.3.3 Measuring the al Properties of a Tower

With a few slight exceptions, a tower gets the same number of attacks whether there is one creep in range or many. In GopherTD, creeps move side by side in two lines. If the lines are in range at the same time, the tower fires no more times than if there is only one. For al, all we need to know is if, for a given time period, there is at least one creep in range. For this reason, when analyzing al, we first transform the set of creeps into a line, where length (in the direction the line is traveling) is important but thickness is not. The density of the set of creeps in range is irrelevant.

To make this process easier to explain, we introduce the following constants:

Creep Length	1 al 0.8 tiles per al
Creep Speed	1 al 0.46 seconds per al

Line Length	14 al 11.20 tiles
From Tile	1.25 al per tile 0.575 seconds per tile 1.739 tiles per second

In Figure 6.8, the usable range of the U-turn in the bottom left is 30 tiles. This includes tiles for both paths; the inner path is 13 tiles long, the outer path slightly longer at 17 tiles. The temporal per-creep size of a tower on the U-turn (Figure 6.8, right) is 21.25al (17 tiles * 1.25 al per tile). The tower's active time for that attack window is 9.78 seconds (21.25al * 0.46 seconds per al) assuming a single creep.

A temporal attack window begins when a tower starts firing and ends when it stops. Since these are caused by the entry of the first creep and exit of the last, when dealing with a set of creeps, we need to know when the first creep enters the attack window and the last creep leaves it. The size of the window is the time it takes the first and last creeps to move through the window plus the amount of time it takes for the last creep to enter the window at the moment that the first creep is leaving it.

```
temporalGroupSize = time it takes first creep to move through   +
                    time it takes last creep to get to entrance +
                    time it takes last creep to move through    -
                    1
```

Figure 6.9 shows the components of the calculation pictorially.

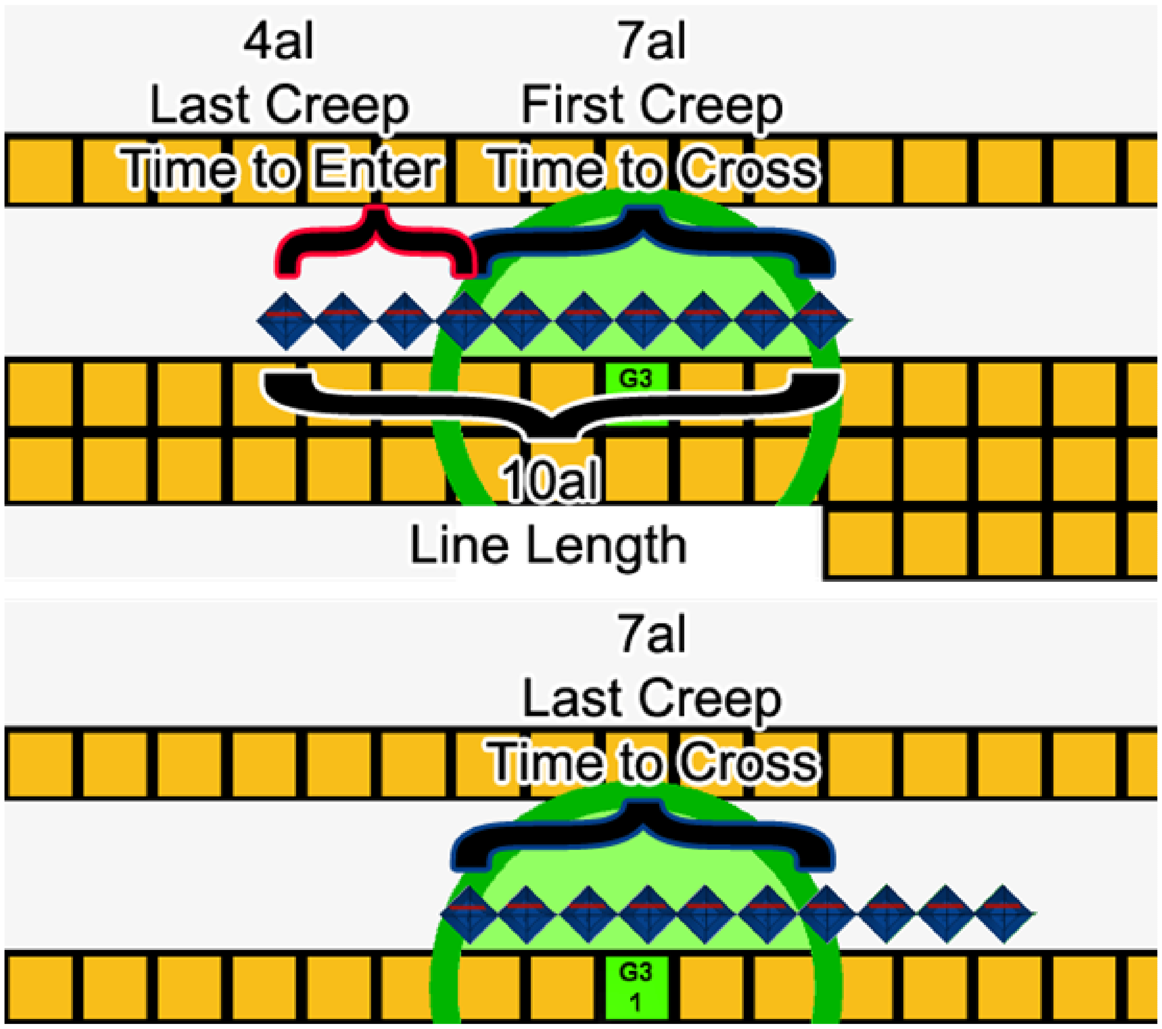


Figure 6.9. Parts of an AL measurement of group attack window size. The total size of the window is the time it takes the first creep to cross the tower's range followed by the time it takes for the last creep to enter the range followed by the time it takes for the last creep to exit the range. TOP: Time for the first creep to move from the attack window's entrance to its exit is 7al. The time for the last creep to enter the attack window from this point is 4al. BOTTOM: As with the first creep, the time it takes for the last creep to move from the attack window's entrance to its exit is 7al. The total group size of the attack window is (2 * 7) + 4 -1 = 17al.

The time it takes for the last creep to enter and then cross the attack window is equal to the length of the line, allowing us to simplify the formula to:

```
temporalGroupSize = perCreepLength + (lineLength - 1)
```

In Figure 6.10, the time it takes one creep to move through the window is 9al and the line is 14al long so the attack window's temporal group size is:

```
    temporalGroupSize = 9al + (14al - 1) = 22al
```

In the shown attack window, the tower gets to attack for as long as it takes 22 creeps to move through its range (in this case, 10.12 seconds; at 10 attacks/second, 101 attacks).

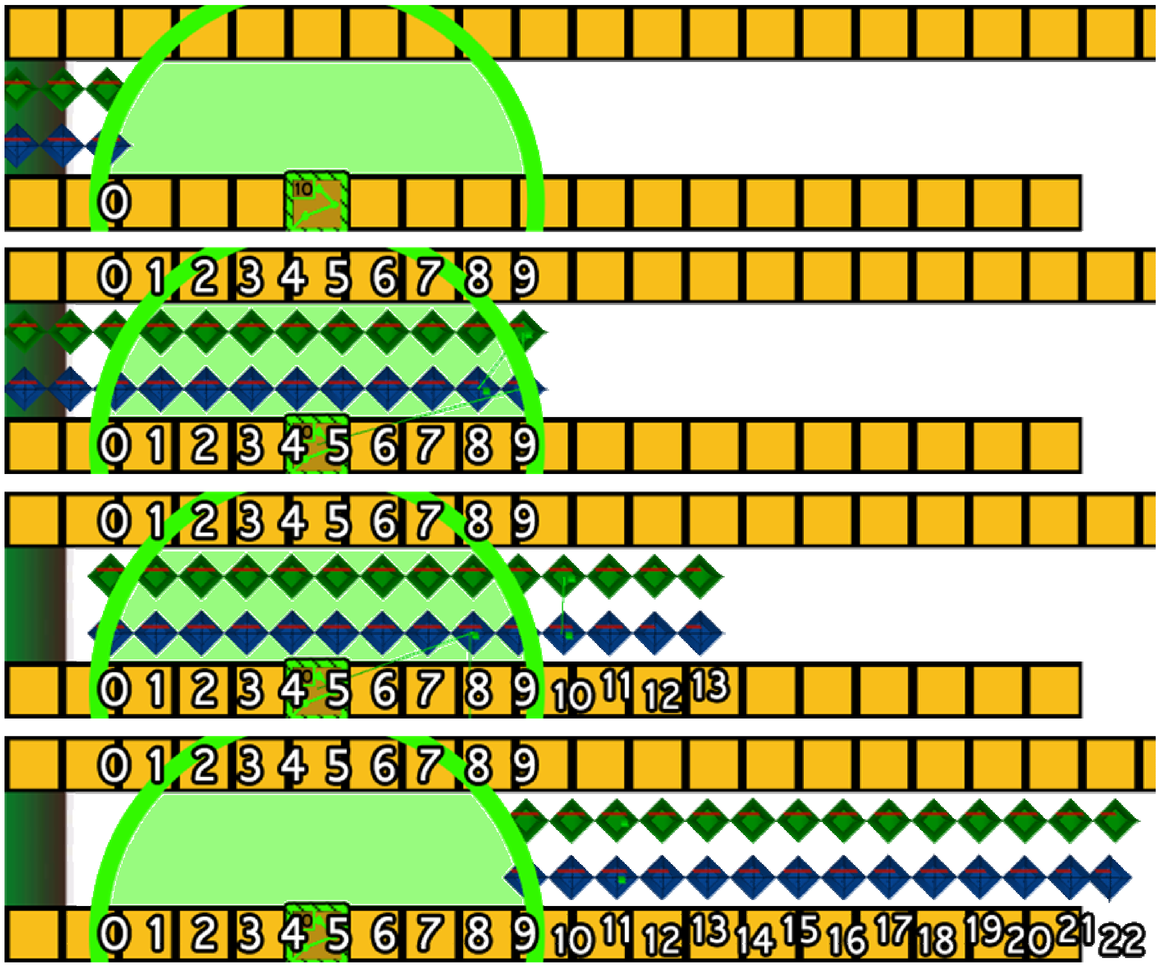


Figure 6.10. Measuring the Size of an Attack Window. TOP: The first creep enters a tower's range. SECOND: The first creep reaches the end of the tower's range, which is 9 creeps long. THIRD: The last creep enters the tower's range. The tower has been active since the first creep entered, 13al ago. BOTTOM: The last creep exits the tower's range.

6.3.4 Effects of Slowing Towers on Measuring Window Size

GopherTD, like most tower defense games, has towers capable of modifying a creep's speed. The calculations discussed here assume an unmodified (i.e., non-slowed) creep. Adding slowing might or might not complicate calculations. Ultimately, we are trying to determine which of a set of positions affords a tower the most active time. This means we need to rank the positions by the amount of time afforded. If an offensive tower is being placed and the slowing towers can be placed around it in a similar fashion at each location, slowing will not affect the ranking of the positions. If all creeps are kept slow in the tower's range, slowing merely scales the magnitude of active time, not affecting the ranking in any way and thus not affecting the position selected.

Unfortunately, slowing is not always applied equally to all creeps. One reason for this is resources. Since a slowing in GopherTD slows a maximum of four creeps, to slow 28 creeps, at least seven slowing towers are needed for each offensive tower being supported. Placing offensive towers near each other allows them to share slowing towers but almost always increases the size of the area to cover and therefore, given the small range of slowing towers, requires additional slowing towers. A second, related reason is that there isn't always room to place seven or more slowing towers.

If all candidate positions have spatially equivalent configurations (ignoring rotation), the placement of slowing towers relative to the offensive tower they support is the same and thus does not affect the ranking of positions. If the local neighbourhood of the candidate positions are not configured similarly, slowing towers might need to be placed in a different configuration which could change how slowing occurs. Relevant variables include the order that creeps are slowed, the degree to which slowing can be reinforced (slowing wears off after two seconds, requiring the creeps to be re-slowed), gap areas (areas where a creep might travel that is out of every slowing tower's range) and tower distribution (towers attack the creep closest to them so slowing towers too close to each other tend to slow the same creep, causing some creeps to be slowed by multiple towers at the same time while others are never slowed, even when seven or more slowing towers are present).

Many subjects centralized all their towers into a central kill zone while others spread their towers out, forming multiple kill zones. Slowing often disrupts the density of creeps – some creeps overlap and merge into short, dense lines while others get separated, creating long, unbalanced lines. Inside a kill zone, this is generally unimportant as the creeps enter in a known order and line length but if the subject has multiple kill zones, disruptions to line length and density inside one kill zone affect how the creeps enter subsequent kill zones.

Finally, although slowing towers have a relatively small range, the range is normally large enough to reach path segments outside the kill zone, causing unintended slowing.

There are several ways for subjects to handle these issues. Issues of gaps in slowing can be dealt with by using more slowing towers than might appear necessary. To avoid line disruptions from compounding between kill zones, a single kill zone can be used. Some subjects also placed slowing towers in areas between kill zones to restore some order to the lines (dense clusters spread out as subsets of creeps are slowed, sparse clusters are compressed by causing leading creeps to wait for others to catch up). To avoid unintended slowing, many subjects explicitly avoided positions that would cause slowing prior to the attack window (subjects didn't worry about unintended slowing that happened after the final kill zone). In many cases, however, the subject was unable to completely predict the effects of slowing and solved the problems empirically; if the subject had trained on the map, they remembered the issues caused by slowing and adjusted their solution, often reactively and atheoretically, to compensate.

As for computer agents, how do we modify the group attack window measurement to account for subsets of creeps being slowed and the line length being disrupted? Our computer agents use lessons learned from al analysis to implement temporal strategies but they do not have the low-level sophistication to account for all effects created by slowing towers. Our agents do not currently implement an executive monitoring process that monitors problems in the execution of a set of strategies (such as unintended effects of slowing) and repairs the strategy set in response. This is an issue for future research.

Having said that, we do not believe that the issue seriously compromises our agent's problem solving abilities. Slowing causes al estimates of window size to be, at worst, underestimates (not counting severe compression between kill zones, which we did not encounter). The performance will therefore be at least as good as predicted by AL measurements. It is possible that differences in positions could affect the ranking of those positions but we did not see any differences between slowing at different positions that we believe were large enough to have a large impact on scores. We also note that humans appear to be no better than the computer agents at predicting complex, unintended impacts of slowing beyond avoiding positions that slow creeps before they enter a kill zone, a capability that we can (but have not yet) implement relatively easily. Even so, the ability to more accurately predict the effects of small changes in slowing towers remains an open and interesting question.

6.4 Quantitative Analysis of Spatio-Temporal Affordances

In this section we look at the al values of popular spatial affordances such as corners, U-turns and path gaps. We also take another look at attack windows and discuss new properties relevant to strategies.

6.4.1 Usable Range

If a tower covers more usable area, will it have more time to fire? The answer seems intuitively true.

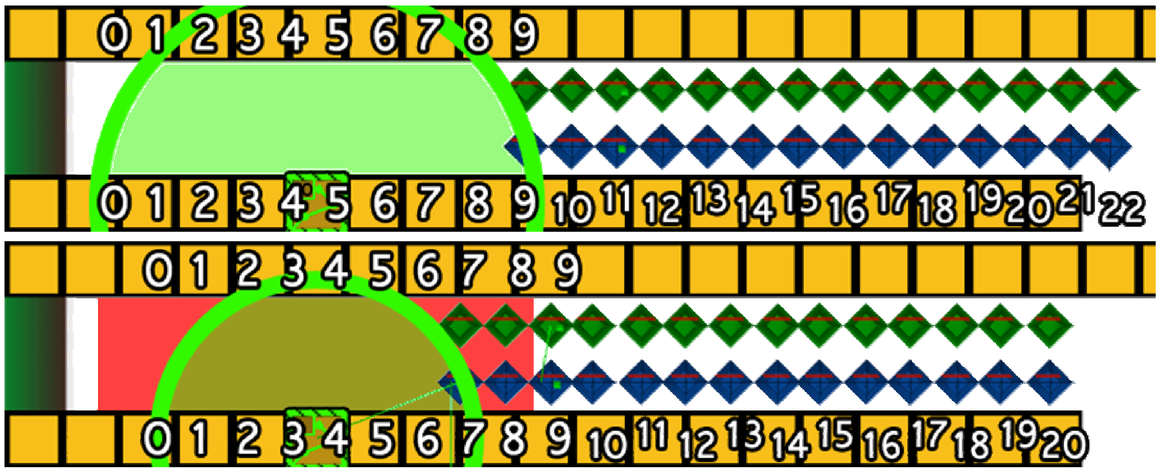


Figure 6.11. Range Size Comparison for Different Upgrade Levels. TOP: A level 10 Green3 tower's range has a size of 22al. BOTTOM: A level 1 Green3 tower's smaller range has a size of 20al.

Consider two Green3 towers, one level 10, one level 1. Green3 towers fire 10 times a second.

	G3-10	G3-1
Range	97 units	70 units
Usable Range	14 tiles	9 tiles
Window Size	9 al	7 al
Line Length	14 al	14 al
Temporal Group Size	22 al	20 al
Active Time	10.1 seconds	9.2 seconds
Shots Fired	101 shots	92 shots

Table 6.3. Spatio-temporal, Temporal and Performance Data for an Upgraded Level 10 Tower vs. a Level 1 Tower.

As expected, the larger the range, the more time the tower has and the more shots it can fire. What is possibly surprising is that the level 10 tower's range in Figure 6.11 covers 50% more usable range than the level 1 tower (usable range=14 vs. 9) but is only 10% larger temporally (22al vs. 20, 10 seconds vs. 9).

6.4.2 Path Gap

In an earlier section we talked about path synchronization gaps, where one line of creeps falls behind the other. This can be caused by slowing towers and any spatial feature of the map that causes one path to be longer than another.

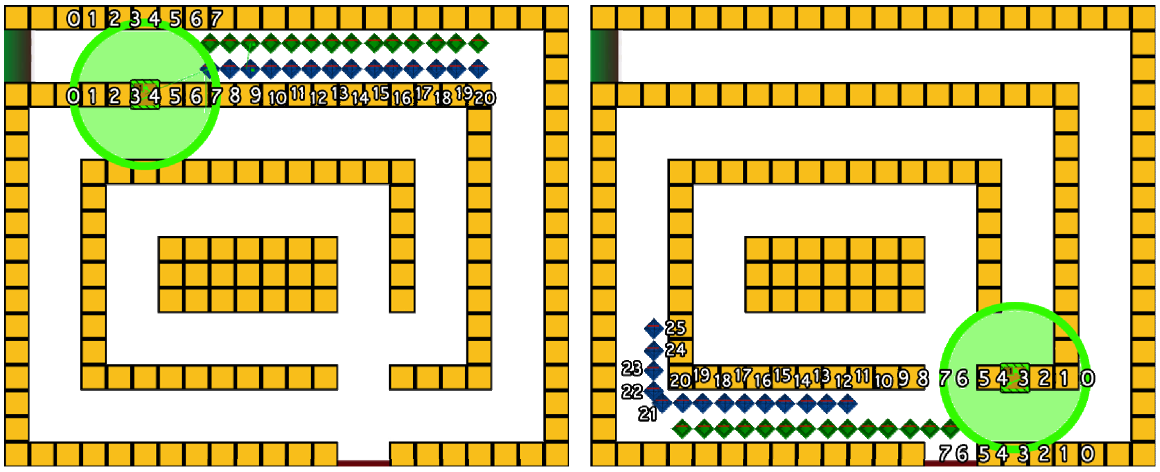


Figure 6.12. Effect of Corners on Path Synchronization. LEFT: A level 1 Green3 tower at the start of a map. al values are shown. RIGHT: A level 1 Green3 tower after two right-hand turns. al values are shown. The outside path is longer and the creeps on it fall 5al behind.

Figure 6.12 shows a tower placed near the start of the map and after the creeps have made two right-hand turns. When rounding a corner, the outside path is longer than the inside path, causing the creeps on the outside path to fall behind. After two turns in the same direction, the outside line falls 5al behind.

The 5al path synchronization gap allows the tower to fire for an additional 5al (2.3 seconds), increasing the number of shots the tower makes by 25%. In Table 6.3, upgrading the Green3 tower from level 1 to level 10 only increases the number of shots by 9.8%. While there are other reasons to upgrade a tower (the level 10 Green3 tower does 405% more damage), from the perspective of active time, increasing the gap between the lines of creeps is significantly more valuable than upgrading the tower.

	No Path Gap	5al Path Gap
Range	70 units	70 units
Usable Range	9 tiles	9 tiles
Window Size	7 al	7 al
Line Length	14 al	19 al
Temporal Group Size	20 al	25 al
Active Time	9.2 seconds	11.5 seconds
Shots Fired	92 shots	115 shots

Table 6.4. Effects of Path Gap. Spatio-temporal, temporal and performance data for a tower placed where the two lines of creeps are still in sync vs. the same tower placed where one of the two lines of creeps has fallen behind.

6.4.3 Corner

Many subjects preferred corners to normal wall positions (i.e., where the path was straight). For purposes of comparison, we will assume the use of a level 1 Green3 tower (a tower with a small range) on either a single-sided exterior corner or a single-sided wall position. Two-sided walls, the most common path-adjacent wall type in GopherTD, are discussed later. Each of these positions has a single attack window, as shown in Figure 6.13.

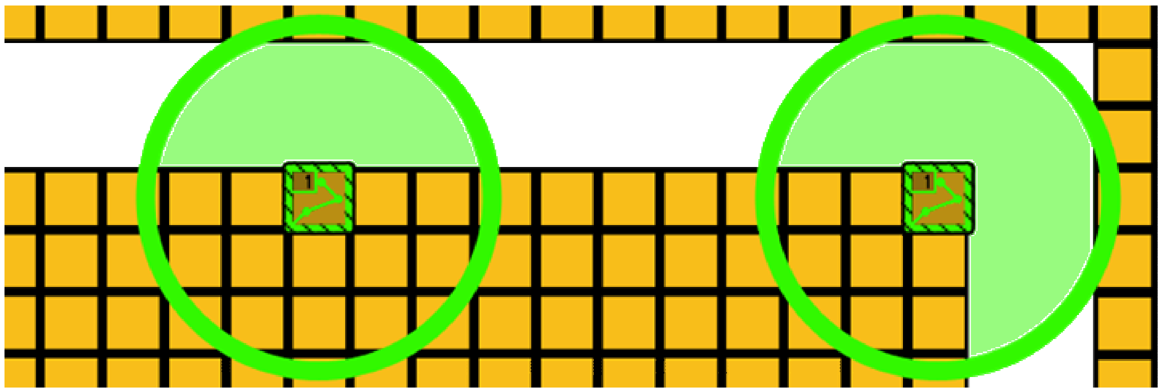


Figure 6.13. A Normal, Single-Sided Wall Position and a Single-Sided Exterior Corner.

Compared to a tower on a normal wall position, a tower placed on a corner covers significantly more usable range. For the level 1 Green3 towers in Figure 6.13, the difference is 14 tiles vs. 9, a 56% gain. The actual growth in the attack window's per-creep size is not quite as impressive – 9al vs. 7al, a 29% gain.

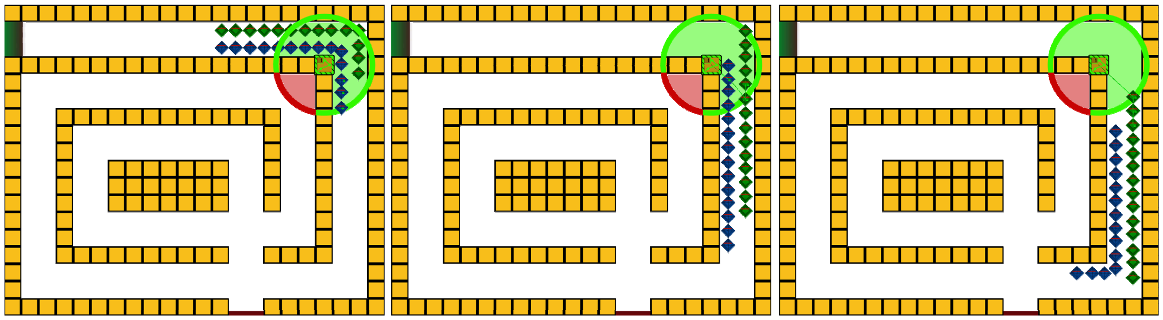


Figure 6.14. Multiple Attack Windows. Two lines of creeps move around a corner. The line on the outside path falls 2.5al behind, stretching the line length to 16.5al. Area in green is the first attack window, area in red is the second attack window, which has a different window size and line length and therefore different temporal group size.

In addition to the corner's size advantage, corners also have a path synchronization gap advantage. If the two lines of creeps enter the tower's range side-by-side at the same time, the outer line fall behind 2.5al, increasing the size of the temporal group attack window by 18% (assuming there are 14 creeps in each line). Technically, the gain is only 1.5al / 11% because of the curvature of a level 1 Green3's range circle (the outer line leaves the circle 1al earlier than the inner line) but to keep calculations general we will ignore the curvature effect for towers with small ranges.

	G3-1 on Wall	**G3-1 on Corner**	**G3-1 after Corner**
Range	70 units	70 units	70 units
Usable Range	9 tiles	14 tiles	14 tiles
Window Size	7 al	9 al	7 al
Line Length	14 al	16.5 al	16.5 al
Temporal Group Size	20 al	24.5 al	22.5 al
Active Time	9.20 seconds	11.27 seconds	10.35 seconds
Shots Fired	92 shots	112 shots	103 shots

Table 6.5. Effect of Corner on Path Gap. Spatio-temporal, temporal and performance data for a tower placed where the two lines of creeps are still in sync vs. the same tower placed where one of the two lines of creeps has fallen behind. All positions are assumed to be single-sided.

As shown in Table 6.4, a tower placed on a corner does 9% better than a tower placed after the corner because of the increased per-creep window size and 21% better than a tower placed on a normal single-sided wall because of the increased path synchronization gap and per-creep window size.

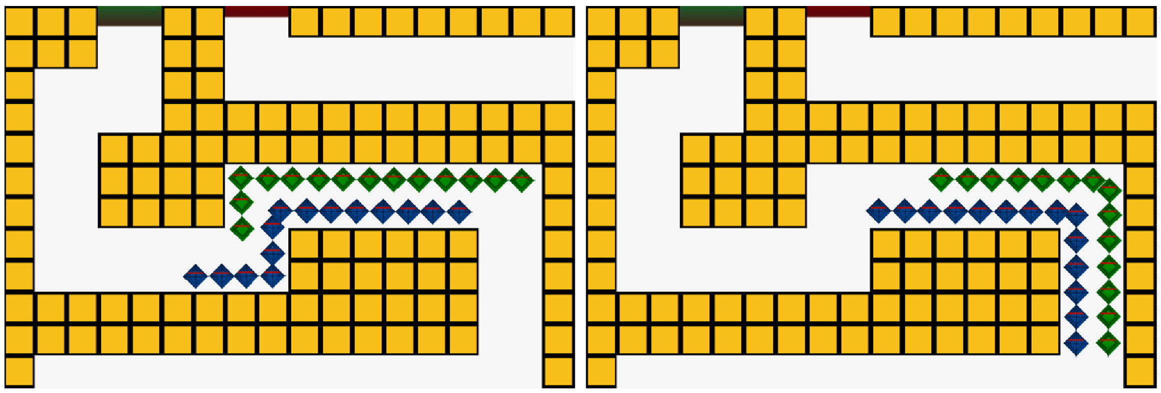


Figure 6.15. Counter-Balancing Corners. LEFT: After two right- and three left-hand turns, the top/green line of creeps is ahead 2.5al. RIGHT: After a third right-hand turn, the two lines of creeps are back in sync.

It should be noted that the above results are based on the two lines of creeps being in synch before rounding the corner. Consider the case where, because of an earlier right-hand turn, the lines are 2.5al out of synch. If the lines make a left-hand turn, the line that used to be the outer line becomes the inner line and moves ahead 2.5al. After the corner, the lines will be back in sync. For this reason, placing a tower after a corner is not always preferable to placing one before it. It is important to know whether the corner will grow or shrink the creep path synchronization gap. Based on observation, it seems that humans either don't notice or are not good at determining this for later corners on maps with multiple turns, although many do appear to accurately remember the path gap for corners they have tried before.

6.4.4 U-Turn

Many subjects preferred to place towers on U-turns. For purposes of comparison, we will assume the use of a fully upgraded level 10 Green3 tower (non-upgraded green towers cannot cover both paths on each side of a two-tile wide U-turn). Figure 6.16 shows two towers on a U-turn, one right at the end of a U-turn, the other one tile back from the end. Some subjects noticed that when a level 10 Green3 tower is placed at the very end of a U-turn, the far end of the range goes into the wall, providing no value, and that this can be solved by moving the tower back one tile. Many other subjects did not notice this. We analyze the temporal group size of the attack window for both.

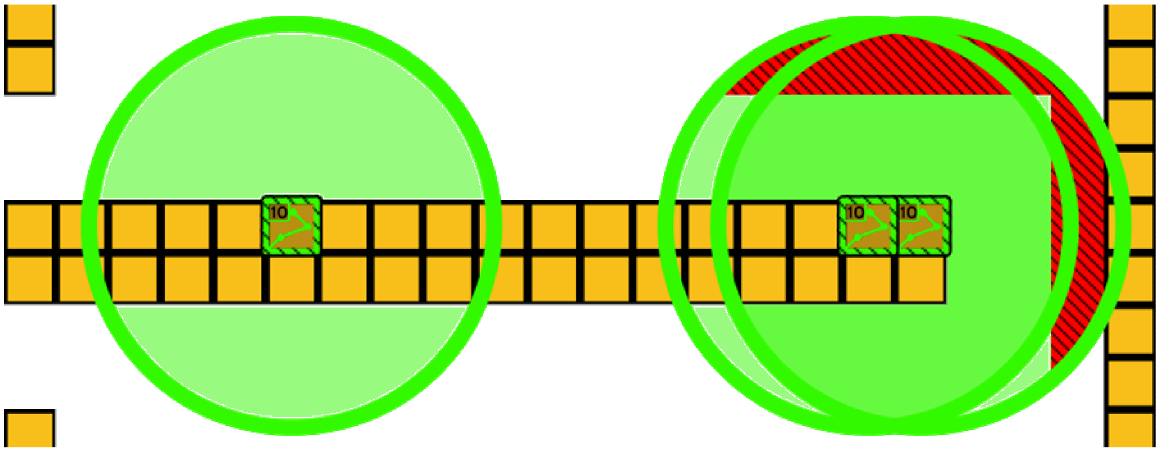


Figure 6.16. U-Turn vs. Double-Sided Wall. Three level 10 Green3 towers. The first is on a normal wall position. The other two are on a U-turn, with one at the very end of the U-turn and the other one tile back from the end. Area in red stripes is in the towers' range but is not traversed by the creeps. Covering this area is no more valuable than covering a wall.

Placed on a corner, a level 10 Green3's range covers 20 tiles and has a per-creep size of 13.5al. The same tower placed at the end of a U-turn or one tile back is 27 tiles / 15al and 30 tiles / 17.5al respectively. Because a U-turn covers two turns in the same direction, it produces twice as much path synchronization loss as a corner. With 14 creeps going past it, the line length of a corner and U-turn are 16.5al and 19al (2.5al per turn) respectively.

	G3-10 on Corner		G3-10 One Back		G3-10 on U-Turn	
Range	97	units	97	units	97	units
Usable Range	20	tiles	30	tiles	27	tiles
Window Size	13.5	al	17.5	al	15	al
Line Length	16.5	al	19	al	19	al
Temporal Group Size	29.0	al	35.5	al	33.0	al
Active Time	13.34	seconds	16.33	seconds	15.18	seconds
Shots Fired	133	shots	163	shots	151	shots

Table 6.6. U-Turn vs. Corner. Spatio-temporal, temporal and performance data for a level 10 Green3 tower placed on a corner, one space back from the end of a U-turn and at the very end of a U-turn.

The U-turn covers more usable area. While this is not directly related to an attack window or tower's temporal group size, it is the easiest quantity for humans to estimate. The assumption most subjects seemed to make was that the tower that covered the most usable area would have the highest active time. The U-turn also has a larger per-creep size and a larger line length. Because these are the two components of temporal group size and temporal group size is directly related to active time, the U-turn leads to more active time (+23%) than a corner, all other things being equal.

6.4.5 Two-Sided Wall

In the earlier analyses we used a normal wall position as the baseline. In doing so, we looked only at the first attack window. Most walls in GopherTD are two tiles thick, allowing any tower to attack on both sides of the wall. Like corners and U-turns, a tower on a normal wall position covers multiple path segments (two for the normal position and corner, three for the U-turn). Unlike corners and U-turns, the path segments for towers at normal positions are not contiguous (see Figure 6.16 for an example).

Most subjects believed that a tower on a normal wall segment covered less usable area than a tower on a corner or a U-turn, even taking both attack windows into account. This intuition is partially correct, as shown in Table 6.7.

	G3-10 on Corner	G3-10 U-Turn	G3-10 AW1	G3-10 AW2	AW 1+2
Range	97 units	97 units	97 units	97 units	97 units
Usable Range	20 tiles	30 tiles	14 tiles	12 tiles	26 tiles
Window Size	13.5 al	17.5 al	9 al	8 al	17 al
Line Length	16.5 al	19 al	14 al	19 al	16.5 al

Table 6.7. Spatial Window Size, Corner vs. U-Turn vs. Two-Sided Wall. Spatial data for a level 10 Green3 tower placed on a corner, one space back from the end of a U-turn and at a normal wall position that covers both sides of the wall (i.e., has two attack windows). For the normal position, statistics are shown combined and for individual attack windows.

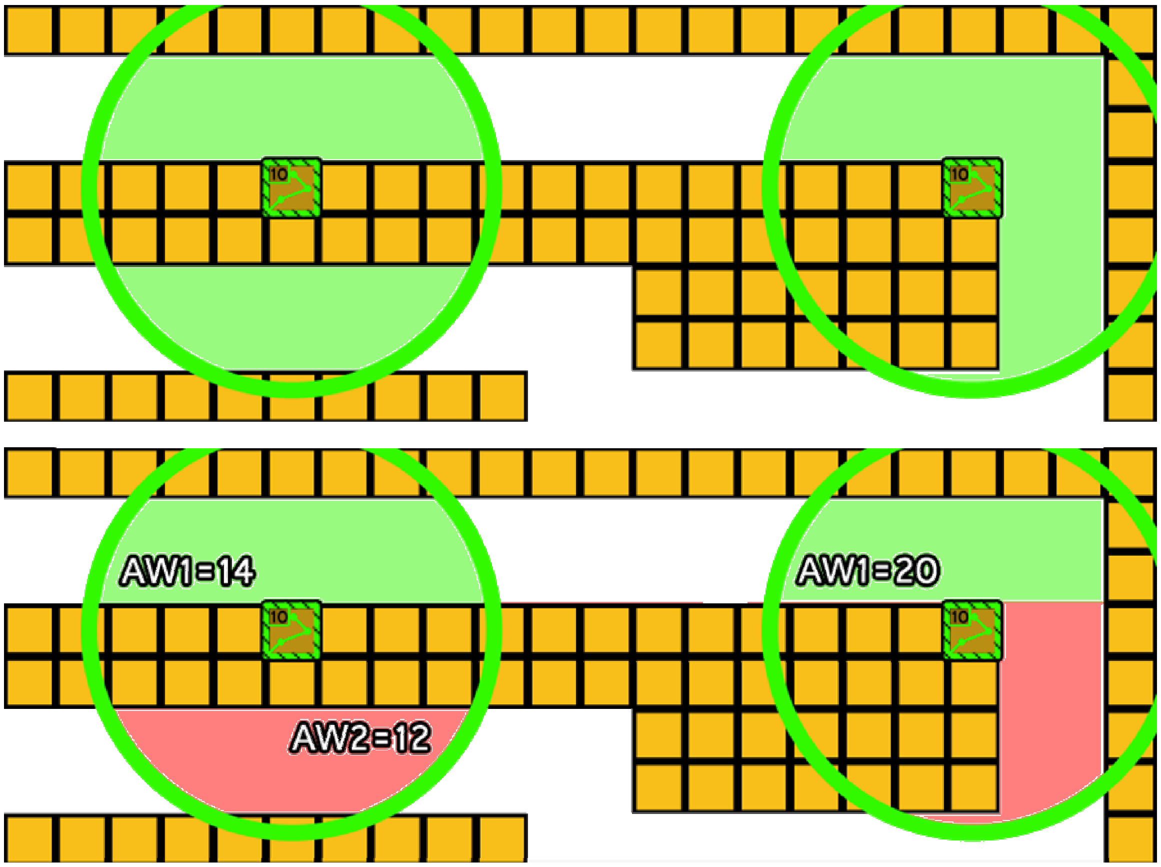


Figure 6.17. Spatial Attack Window Size, Two-Sided Wall vs. Corner. TOP: The usable range of a tower at a normal position and on a corner. Many subjects believed that the corner covered more area. BOTTOM: Usable range for each position. Green area represents the first attack window (left) or equivalent area (right). Red refers to the second attack window.

Spatially, a tower on a normal wall segment is inferior to a U-turn on all criteria but as good as a corner on some criteria (average line length) and better on others (combined usable range and per-creep window size). Perhaps subjects believed that corners covered more area because the corner's single attack window is 43% larger than the largest of the normal position's attack windows. Note that, in Figure 6.17, the corner position's usable range would be larger if the tower were moved one tile back from the right but it would only increase from 20 tiles to 21, well below the 26 tiles covered by the normal position.

Given that the normal position has a larger combined per-creep window size and the same line length (averaged equally over the two attack windows), it is intuitive that the normal position leads to a higher active time.

Likewise, it is intuitive that a tower on a U-turn would have a higher active time than a tower on a normal, two-sided position. The U-turn covers 15% more usable range, which appears to be the primary determinant for humans to determine that it is a better position. The U-turn also has a higher window size (3%) and line length (15%), the two determinants of temporal group size. It is therefore surprising that the U-turn does not, in fact, have a higher active time.

The normal position, the one likely picked if no strategy is used, is inferior in all spatial measures and each attack window, individually, is inferior in every category. If we analyze the positions from the point of view of a single creep, the U-turn is clearly superior to the normal position. When analyzed from the point of view of a set of creeps, however, the answer is quite different.

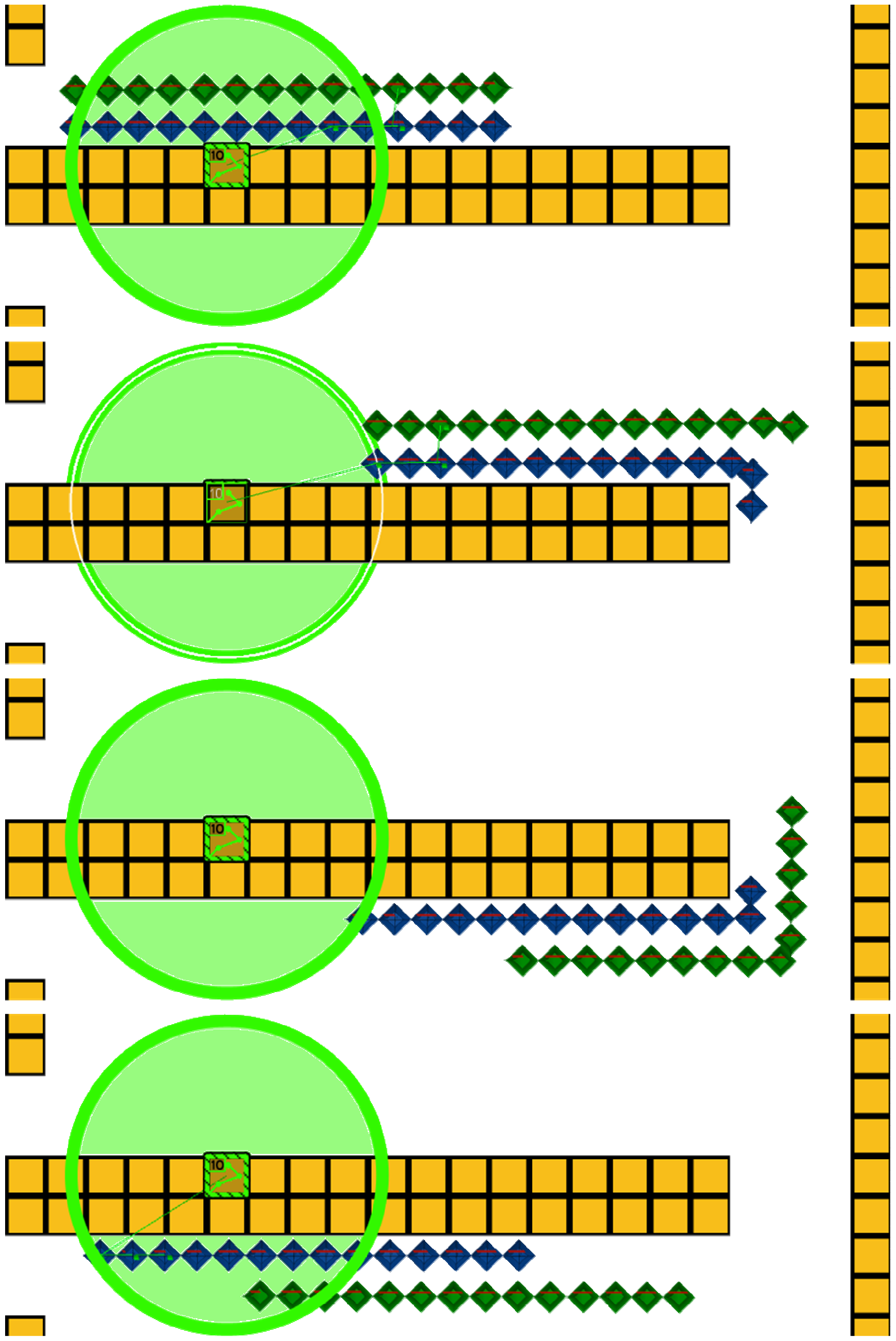


Table 6.18. Movement Past a Two-Sided Wall Position. TOP: The creeps move through the tower's first attack window. BOTTOM: The creeps come back, moving through the tower's second attack window.

	G3-10 on Corner	G3-10 U-Turn	G3-10 AW1	G3-10 AW2	AW 1+2
Range	97 units	97 units	97 units	97 units	97 units
Usable Range	20 tiles	30 tiles	14 tiles	12 tiles	26 tiles
Window Size	13.5 al	17.5 al	9 al	8 al	17 al
Line Length	16.5 al	19 al	14 al	19 al	16.5 al
Temporal Group Size	29.0 al	35.5 al	22.0 al	26.0 al	48.0 al
Active Time	13.34 sec	16.33 sec	10.12 sec	11.96 sec	22.08 sec
Shots Fired	133 shots	163 shots	101 shots	119 shots	220 shots

Table 6.8. Temporal Window Size, Corner vs. U-Turn vs. Two-Sided Wall. Spatio-temporal, temporal and performance data for a level 10 Green3 tower placed one space back from the end of a U-turn and one at a normal wall position that covers both sides of the wall (i.e., has two attack windows).

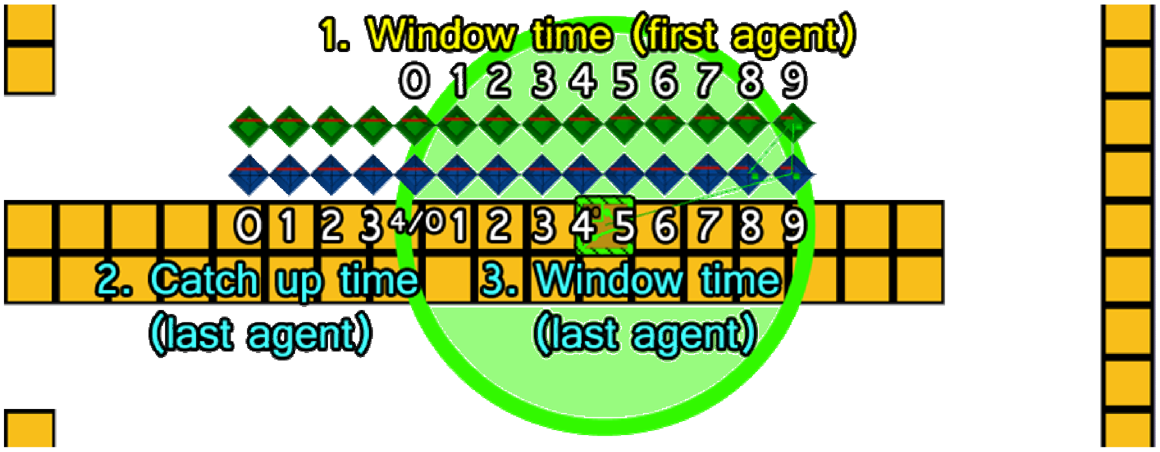

1. Window time (first agent)
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4/0 1 2 3 4 5 6 7 8 9
2. Catch up time (last agent)
3. Window time (last agent)

Table 6.19. Creeps Moving Through a Spatial Attack Window. The temporal group size of the window is made of three parts: (1) The amount of time it takes for the first agent to make it from the start of the range to the end followed by (2) the amount of time it takes for the last agent to make it to the start of the range (from the time that the first agent leaves the range) followed by (3) the amount of time it takes for the last agent to make it from the start of the range to the end. The tower will be active from the time the first agent enters the range until the time that the last agent leaves it.

An attack window's size, as measured by the amount of time it takes a group to move through it (i.e., the temporal group size), is determined by the amount of time it takes the first group member to move all the way through the window followed by the amount of time it takes for the last group member to both make it to the entrance of the tower's range and then make it to the range exit. This process for a tower with two attack windows is shown in Table 6.18.

We can divide the temporal group size formula into two parts – the *spatial attack window time* and the *catch up time*.

The spatial attack window time is the amount of time it takes the first agent to move through the space covered by the tower (the spatial attack window). While the agent does this, the agents behind it continue to move forward. In GopherTD, the tower attacks the agent closest to it and continues to attack it until the agent is destroyed or exits the range. Since the first agent will initially be the only one in range, the tower will focus on that agent. While the tower is focused on the first agent, it cannot attack other agents, allowing the other agents to move safely through the tower's range.

The catch up time is the amount of time it takes for the remaining agents to move through the tower's range (the spatial attack window) once the first agent moves out of range. This is equivalent to the amount of time it takes for the last agent to arrive at the start of the spatial attack window and the amount of time it takes for the agent to then make it across this area. Since the catch up time starts when the first agent of the group moves out of range, this is the tower's first opportunity to attack the remaining set of agents. The larger the spatial attack window, the longer the first agent keeps the tower's attention and the more time the last agent has to make it to the start of the spatial attack window.

```
temporalGroupSize(U-turn) = spatialWindowTime + catchUpTime
   = time for first agent to cross
     the range + time for last agent +
     to enter the range + time for
     last agent to cross the range
   = windowSize + ((lineLength-1) - windowSize) + windowSize
   = 17.5 + ((19-1) - 17.5) + 17.5
   = 35.5
```

In the above example, the U-turn's window size is large but the catch up time is only 0.5al because the last agent is almost inside the tower's range by the time the first agent makes it out.

The key to understanding the issue is this: the longer a single attack window is, the more time the tower spends on the first agent and therefore the more time the remaining agents have to move towards the exit. As the spatial window time increases, the catch up time decreases. If a tower has a small spatial attack window, the spatial window time is smaller but the catch up time is larger. If there are multiple small spatial attack windows, the catch up time can be counted multiple times.

```
temporalGroupSize(AW1) = spatialWindowTime + catchUpTime
   = time for first agent to cross the range +
     time for last agent to enter the range  +
       time for last agent to cross the range
   = 2 * windowSize + ((lineLength-1) - windowSize)
   = (2 * 9) + ((14-1) - 9)
   = 18 + 4
   = 22

temporalGroupSize(AW2) = 2 * windowSize +
                         ((lineLength-1) - windowSize)
   = (2 * 8) + ((19-1) - 8)
   = 16 + 10
   = 26

temporalGroupSize(1+2) = 22 + 26
   = 48
```

For the two attack windows, the combined amount of time for the first agent to move through both of the tower's spatial attack windows (17) is slightly lower than moving through the U-turn's single attack window (17.5) but the time it takes for the remaining agents to move through the spatial attack windows is significantly higher (18 vs. 31). As a result, it takes the set of 14 agents 35% more time to move through the two smaller spatial attack windows than the one larger one. It doesn't take any specific individual agent more time but groups do not move through the time in the same way individuals do.

It is important to note that the above only applies to towers with multiple attack windows that are temporally non-contiguous. If there are agents in one spatial attack window when different agents enter the next (i.e., they are the same temporal attack window), the tower will continue to fire at the creeps in the first window, allowing the other creeps to move through the second one unharmed, behaving the same as if it had been one larger window.

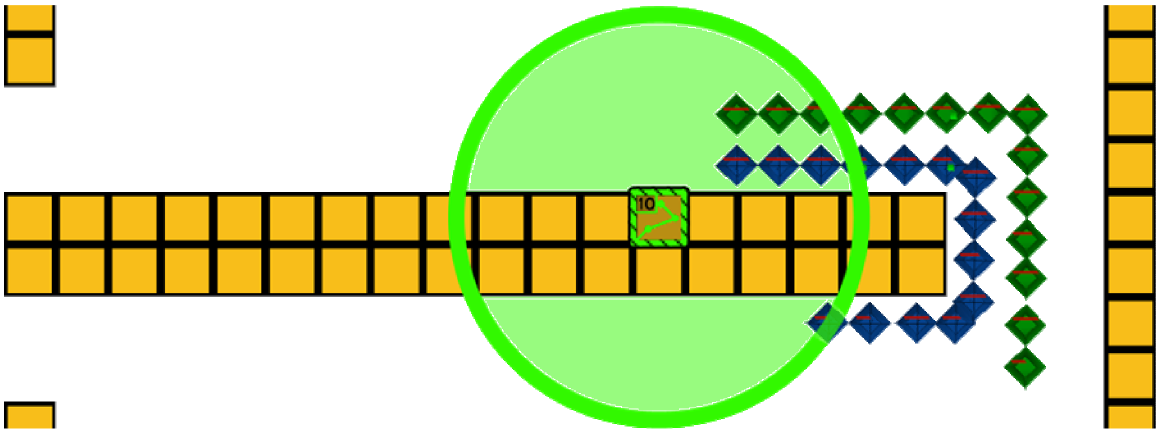


Figure 6.20. Multiple Spatial Windows, Single Temporal Window. There are two spatial attack windows (two disconnected areas) but, because there are still creeps in the first one when the other creeps enter the second, the tower never stops attacking, meaning the attacks occur in a single contiguous block of time.

6.4.6 Buffering and Attack Window Decay

In theory, the size of the per-agent attack window is the same for all creeps, meaning that a tower has just as much time to attack one creep as any other. In practice, this is not true. As time in a group attack window progresses, the per-agent attack window shrinks. We call this *attack window decay*.

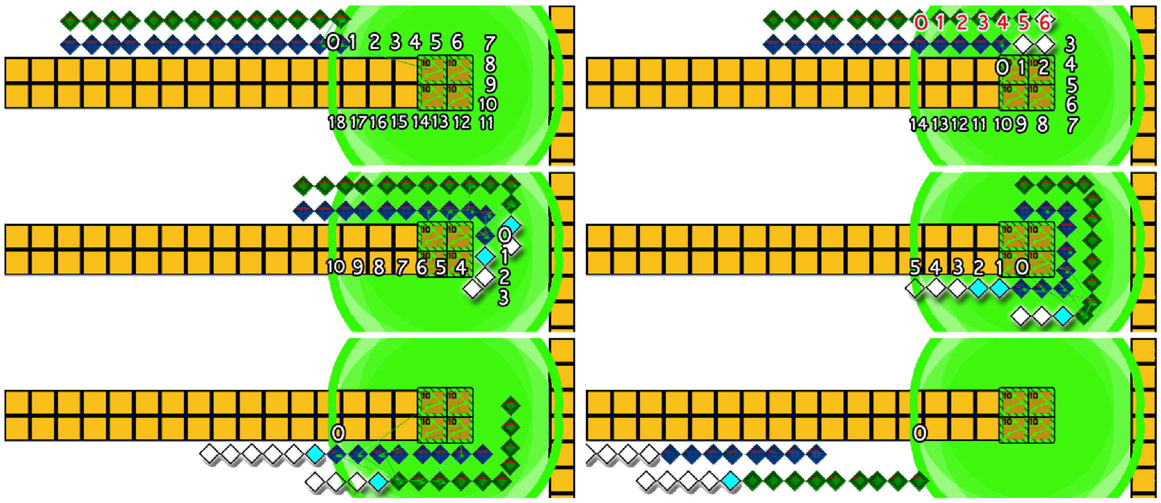


Figure 6.21. Movement Through a U-Turn. The per-agent attack window size starts at 18al but shrinks as time goes on. Red numbers (top right) indicate the amount of time it took to destroy the first agents (6al). White numbers represent the current size of the per-agent attack window. White diamonds represent where agents would have been had they not been destroyed. Blue diamonds are the most recently destroyed creeps.

Creep	Starts At	Destroyed At	AW Size
1	0	6	18
2	4	9	14
3	8	15	10
4	13	19	5
5	18	-	0

Table 6.9. Attack Window Decay. The starting and ending points of the agents in Figure 6.21. Starts At and Destroyed At are positions relative to the starting 18al per-agent attack window. Attack Window (AW) Size measures the remaining amount of the tower's range (in al) that the targeted agent must cross while being attacked. Because Green3 towers attack three agents at a time, statistics are reported for the first agent in an agent set, where an agent set is a group of agents that are destroyed at the same time.

Figure 6.21 shows two parallel lines of 14 creeps moving past a U-turn covered by four level 10 Green3 fast attack towers. From the perspective of the first creep, the combined tower attack range is 18al long, meaning the creeps must move 18al through the towers' range and the towers have 18al (8.28s) to attack the creep. Because towers can only attack a fixed number of creeps at a time (in this case, three), as long as the towers are attacking the first creep, the other creeps in the towers' range can move through the range unharmed. The creep being attacked acts as a defensive buffer for the other creeps in range, a process we refer to as *buffering*.

It takes 6al for the four level 10 Green towers in Figure 6.21 to destroy the first three creeps (all three are attacked at and destroyed at the same time). When those creeps are destroyed, the towers switch their focus to the next closest creeps, which are the next creeps in line. By the time they are targeted, those creeps have moved 4al through the towers' range. The previously targeted creeps had to move 18al through the towers' range while the newly targeted ones only have to move 14al. The towers have 14al (6.44s) to attack these creeps, a drop of 22%. When the second group of creeps is destroyed, the next creeps to be targeted have moved 8al through the towers' range, shrinking the towers' attack window for those creeps to 10al (4.6s), a further 29% drop. This continues until all the creeps are out of range or the attack window size drops to 0 (technically, because the towers select the creeps closest to them, at a certain point they will stop targeting the first creeps in line and start targeting the ones in the middle, providing a lower bound on the per-agent attack window size).

The shrinking of the per-agent attack window does not reduce the size of the group attack window. The negative impact is that it spreads the damage out. The towers in Figure 6.21 require 6al to destroy a creep. After destroying the third set of creeps, the per-agent attack window drops to 5al, making it impossible for those towers to destroy any more creeps. Shorter per-agent attack windows allow the towers to wound more creeps but the score only increases for creeps that are completely stopped.

Chapter 7

Solution Representation

Subjects attempted to prevent a group of objects (creeps) from moving through a map by placing a set of defense towers at strategic locations. Each map is a problem to be solved and each configuration of towers to that end is a *Solution Attempt*. GopherTD automatically recorded subject ID, score, date, time spent placing towers and tower-level data such as type, level, grid position, placement order and performance data (total damage, total overage, average overage, shots fired, overage shots fired).

Higher level data was manually captured by the experimenters using a set of coding sheets we developed. We use *coding* in the psychological sense of recording features on a questionnaire or similar form. Each solution was coded independently by two people to ensure accuracy. Discrepancies between coders were normally due to ambiguity in the coding sheet and were resolved by revising the coding sheets or explicitly defining ambiguous material.

The primary coding sheet used in subject interviews was the *Solution Attempt Tower Configuration Coding Sheet*. It focuses on where each tower group was placed and the motivation for that placement. It changed several times during the study. There were several reasons for this:

- **New Information**. When we began, we felt our coding sheet, devised by two subject matter experts, was quite comprehensive. We were wrong. Subjects frequently mentioned features and concepts during problem solving that we could not capture. New content drove most of the changes made during Phase One.
- **Inter-Rater Disagreements**. Coding is a manual and subjective process. To validate the quality of the work, two coders independently code each solution attempt and the results are compared for inter-rater agreement. To validate the coding process, both joint and independent coding of sample data was done. This uncovered a number of discrepancies, most due to insufficiently precise field descriptions. A number of changes were made to address this.
- **Computational Modeling**. The coding sheet served both as a description of subject solution attempts and, once aggregated, a de facto model of the problem solver's play style. We created computer models of several subjects. In doing so, we learned that the coding sheet/model was deficient in some places. In some instances the problem was that we had not captured sufficient information. In others, it was that subject descriptions did not match their actions, prompting us to ask new questions in interviews and add and revise fields to better capture what subjects meant rather than what they said.
- **Efficiency and Re-Framing**. The process of coding, gaining inter-rater agreement and computationally implementing the models uncovered several places where fields could be combined or re-phrased and notation could be simplified.

It is important to note that all fields on the coding sheets were derived from human subjects. While other values are certainly possible, the coding sheet only includes values we observed subjects using.

The Solution Attempt Tower Configuration Coding Sheet records data at both the tower and overall solution levels. At the tower level, data is captured first by role and then by functional group. Captured for each functional group is the group size, placement decision (target, anchor and relationship), upgrade bias, density bias, chaining bias and any fields specific to that role. The data captured at the solution level includes general goals (offense, placement, damage and creep density), tactics (offense pieces, slowing and offense types), concerns and miscellaneous biases.

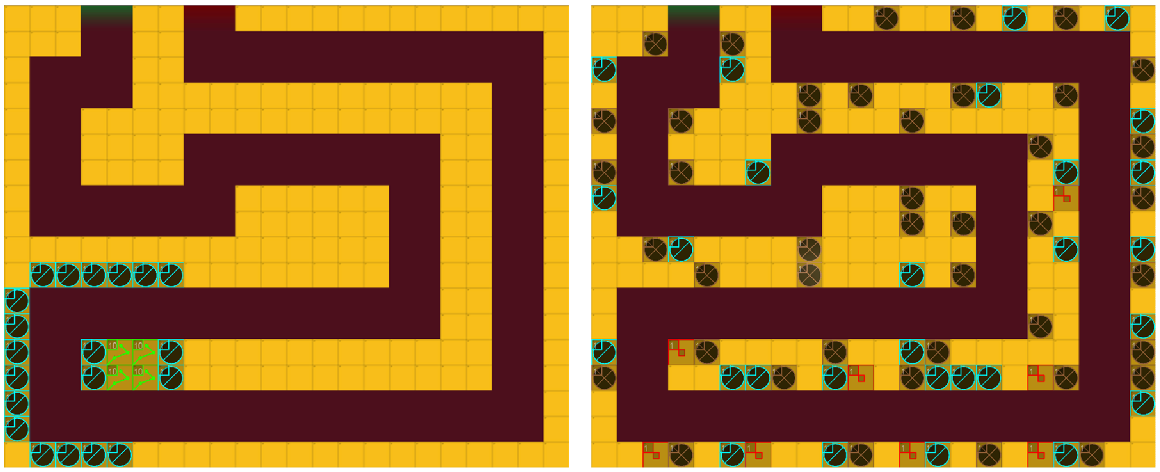


Figure 7.1. Tower Roles. LEFT: Subject E0100 used one Heavy group with fully upgraded fast towers placed on a single U-turn and one Slowing group chained around the Heavy. Roles=Heavy, Slowing. Placement=On U-turn. Upgrade=max (Heavy), none (Slowing). Density=concentrated. Chaining=touching (Slowing). RIGHT: Subject E1700 spaced his non-upgraded offense towers along the bottom of the map and spaced slowing towers all along the path. Roles=Heavy, Slowing. Placement=Middle of path (Heavy). Upgrade=none. Density=spread out. Chaining=distributed over range.

7.1 Role

Each tower plays at least one role, where role means an intended function. Example real-world roles include doctor, police man and construction worker. Towers do not have innate roles; a tower's role is determined by the intention of the person placing it. We determined each tower's role from interviews and think aloud data. A tower may play multiple roles (for an example, see STRATEGY E5: RE-USE HEAVY AS CLEANUP).

There are four roles:

- **Heavy**. Heavy towers are the primary damage dealing towers. They may be strong or weak offense towers. Most solutions have at least one heavy tower.
- **Moderate**. Like heavy towers, moderate towers are intended to damage creeps. They are not, however, intended to be the primary damage dealer. They are support towers. If placed before heavy towers, they are normally used to "soften" (wound) creeps before they enter the heavy's range. When placed near the heavy, they are normally used to deal damage faster or finish off wounded towers in an effort to reduce excess (the amount of damage a tower does that is above the amount needed to stop a creep). They are also used when the player cannot afford another heavy.
- **Slowing**. Although slowing towers can do damage, they are designed to delay their target.
- **Cleanup**. When a creep makes it past the primary attack towers, it is called leakage. These creeps are often significantly wounded. The goal of cleanup towers is to finish off heavily wounded creeps. Cleanup towers are always placed after the primary attack towers (heavy and moderate). Cleanup groups normally include at least one weak offense tower and often include one or more slowing towers.

Almost all solutions included slowing towers. Although all solutions had a tower that caused damage, not all used an offense-oriented tower (explained below). Moderate towers were common but not ubiquitous.

Not all subjects used cleanup towers but many of those who did were often quite emotional about them. We observed significant psychological distress in subjects when a small number of heavily wounded creeps made it past their defenses, leading to the equivalent of buyer's remorse.

While not common, it is possible for a tower to play more than one role. The two most common situations were STRATEGY TP2: SWAMP (slowing towers are used as offense towers) and STRATEGY E5: RE-USE HEAVY AS CLEANUP (heavy towers act as both primary offense and cleanup).

7.2 Functional Group

A functional group is one or more towers trying to achieve the same low-level goal. This goal might be to attack at a particular corner, slow creeps entering a particular tower's range or slow one line of creeps at the map entrance as part of a DIFFERENTIAL SLOWING strategy.

Tower information (placement, upgrade, density and chaining) is tracked at the group level. This follows subject descriptions of solution attempts (e.g., "put a pair of towers at each corner", "put a few towers around the big tower", "put most of my towers along this corridor").

A group can be any size from one tower to all of them. Group size was tracked as single tower, a few, half, most, all and equally split across groups. Size is relative to the role – if a group of slowing towers is of size=all, it refers to all slowing towers, not all towers.

Most subjects used the same number of groups on most maps. A few, however, had strategies such as "place heavy towers at every U-Turn and surround them with slowing towers" that caused the number of groups to vary by map. If groups had the same goal (i.e., attack at a U-turn), subjects often tried to have each group be the same size. Since the budget to purchase towers is fixed, as the number of groups increased, the number of towers in each group necessarily decreased.

Functional groups are determined by goals. Consider the left image in Figure 7.2. When asked about the eight towers at A1, the subject might explain their goal as a group (i.e., keep creeps slow when they're near tower A) and then explain that the towers at B1 are similar to those at A1 (only for tower B). While the only guaranteed way to determine an agent's goals is to ask, for the purposes of grouping it is sufficient for us to recognize when a set of towers has the same placement decision. All eight towers in A1 are placed next to tower A and thus presumably share the same intended goal.

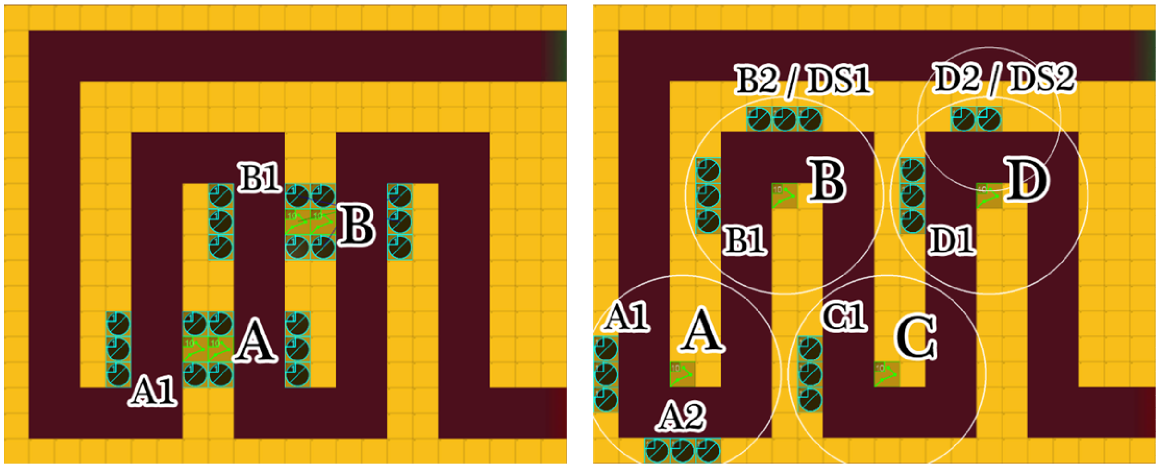


Figure 7.2. Tower Role Functional Groups. LEFT: Subject s0400 used four groups – two Heavy groups (A, B), each surrounded by one Slowing group (A1, B1). RIGHT: Subject 3E0100 used 11 groups – four Heavy groups (A-D), each with one slowing group at its range entrance (A1-D1) and most with one slowing group midway through its range (A2-D2). The groups B2/DS1 and D2/DS2 effectively play two roles. They support the SLOW IN RANGE strategy for Heavies B and D. They also execute the DIFFERENTIAL SLOWING strategy. The latter was unintentional and therefore not counted in the subject's strategy summary.

Placement decisions can be a proxy for low-level goals. Both can be represented at different levels of granularity. In the left image in Figure 7.2, the eight slowing towers in A1 are placed next to tower A or, alternately, inside tower A's range. There is one slowing group per heavy group. In the right image, there are two slowing groups per heavy. Group A1 is a chain placed at the tower A's range entrance while A2 is a chain placed in the center of the range. To a human (certainly to a domain expert), these six towers belong in two separate groups, not one.

A given position can often be explained by more than one placement decision, which can then suggest different goals and thus different function groupings. For example, the left towers in group A1 (Figure 7.2 left) are both in range of tower A and on the end of a U-turn. Determining which explanation is more likely is often possible if the set of strategies is known or the set of goals, from which one can infer strategies. In this case, placing slowing towers near a heavy tower group is consistent with the STRATEGY SLOW IN RANGE while placing slowing towers at the end of a U-turn that does not have offense towers on it is not consistent with any known strategy and achieves no known goal. SLOW IN RANGE is further suggested by the fact that it is a common strategy (almost all subjects used it) and appears in the subject's other solutions. In this way, a subject's strategies can be inferred, although in our work we determined strategies and placement decisions by interviewing subjects and explicitly asking about motivations.

The placements in Figure 7.2, right, are more difficult. The towers in groups B2 and D2 are a good fit for the SLOW IN RANGE strategy. They are also, however, the best positions to use for a DIFFERENTIAL SLOWING strategy. A third option is that they were placed there for both reasons. This subject did not use DIFFERENTIAL SLOWING in any other solution attempt but the only way to confidently determine the subject's intent in this case is to ask.

7.3 Placement Decision

A placement decision says that a certain item will be placed so that it has a certain relationship with a second item. A simple example would be placing a tower next to the path. The three parts of the relationship are called the placement Target, Relationship and Anchor and are represented by the 3-tuple [target, relationship, anchor]. If the subject's only concern was to place a tower immediately next to any path cell, the placement decision would be [target=Tower, relationship=Touching, anchor=Path].

Ultimately, the only action the problem solver has is to place a tower on a grid cell. This is not, however, how the problem solver views the problem. They think in terms such as placing a tower's range on top of a U-turn or placing a tower where it can help a second tower.

From the human data we have identified seven targets, 19 relationships and 37 anchors. A placement decision has one target and one relationship but can have multiple anchors. Anchor combination relationships are explained below. An analysis of the number of unique anchor combinations and placement decisions is given in Section 7.7 Representational Complexity.

7.4 Target

The placement target is the item being positioned on the map. While the problem solver must place a tower in a grid cell, they are often attempting to place something else. The most common placement target is the tower's range area, denoted by a circle in GopherTD. The problem solver will often attempt to place this circle so that it covers as much of the path area as possible.

It is often possible to tell from the attempted solution what the problem solver was using as a target. In some cases, however, the problem solver makes identical placements but switches from one target to another. Specifically, subjects might find an easier way to represent a placement. Several subjects had the goal of maximizing the amount of path area covered by a tower. Initially, they attempted to place a tower's range, moving it across the map to determine where it covered the most path area. This often led to placing towers on corners and U-turns. On later maps, rather than comparing tower range to path area, they immediately placed towers on corners and U-turns. The goal was the same (maximize usable range) but the placement target had changed (tower range vs. tower).

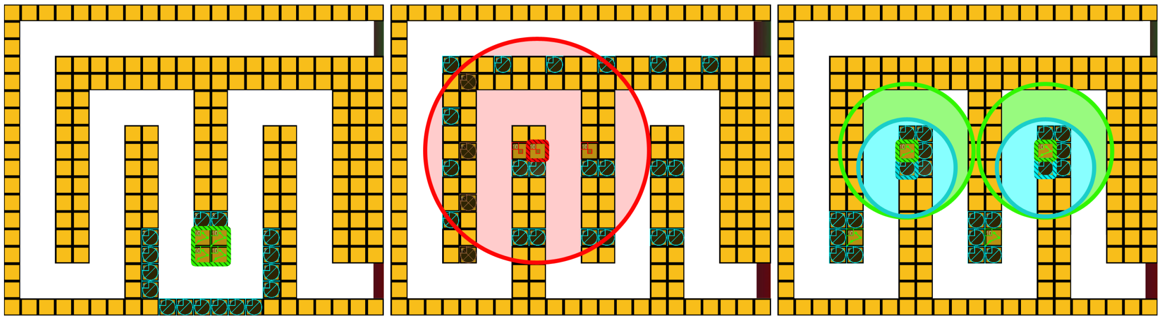


Figure 7.3. Placement Targets Tower, Range and Range Entry. LEFT: Tower (E1100). Places towers at the end of U-turns. CENTER: Range (e2600). Places tower's range where it covers the most path area. RIGHT: Range Entry (3e900). Places the slowing tower's range's entrance so that it lines up with the offense tower's range's entrance.

Values observed in the human data:

- **Tower**. The subject focused on the tower specifically rather than a property of the tower. Often done when placing towers on corners, along the path and inside chains and other group patterns.
- **Range**. The subject focused on where the range would be placed. Often done to cover the most path area or number of passes. This was the most common option.
- **Range Entry**. The subject focused on the point where the creeps enter a tower's range. There are two common reasons for this. For offensive towers, a set of towers in a functional group would align entry points so that they selected the same target (the FOCUSED FIRE strategy). For slowing towers, a slowing tower would align its range entrance with an offensive tower's range entrance so that the slowing tower slowed creeps just as they entered the offensive tower's range. This is important because if the creeps were slowed too late, attack time would be lost since the creeps would move through part of the range quickly and if they were slowed too early, the creeps behind the first (now slowed) creeps would have time

to catch up, causing the creeps to clump and undoing any separation accomplished earlier.

- **Range Exit**. The subject focused on the point where the creeps leave a tower's range.
- **Attack Window**. The portion of a tower's range area that covers the path is called the usable range. If the path leaves the range and then re-enters it, the tower ends up with multiple, disconnected usable range areas. These are called attack windows (a more precise definition was given in 6.2.2 Attack Windows). A subject may choose to focus on placing one of these attack windows. Often used with RE-USE HEAVY so that the tower's range covers the map exit.
- **Pair of Attack Windows**. The subject focuses on the relationship between two attack windows. This is typically done when executing the ATTACK WINDOW SEPARATION strategy, which attempts to prevent creeps from entering one attack window until they have fully exited the other.
- **Slowing Effect**. The subject focuses on the status of a creep at a certain position. Slowing towers cause affected creeps to move at half speed for two seconds. The area in which a creep is slowed begins inside the tower's range but can extend beyond it.

7.4.1 Relationships

Relationships exhibit three properties, direction, distance and precision. Direction is used to say that the placement target is before (around, within, etc.) the placement anchor. Distance is used to say how close the target should be (near, at a distance from, etc.) to the anchor. Precision describes how picky the problem solver is about distance.

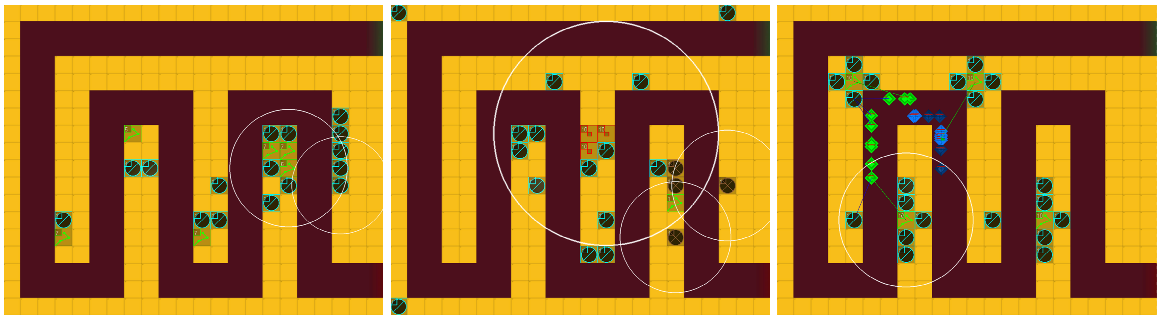


Figure 7.4. Placement Targets Range Exit, Attack Window, Pair of Attack Windows. LEFT: Range Exit (E1300). Subject places the slowing tower's range's exit so that it lines up with the offense tower's range's exit. CENTER: Attack Window (a0100). Subject places the red offense tower's last attack window so that it is near the path exit. RIGHT: Pair of Attack Windows (3E0600). Subject divides the tower's range into two attack windows and positions them so that the creeps almost completely leave the first window before entering the second.

Subjects varied in how specific their placement criteria were. Some subjects demanded that a group of towers be touching while others were satisfied if the towers were simply near each other. Less precise relationships typically have more positions to choose from since they include all the positions from the more precise relationships plus, potentially, others.

Precision sometimes varied within the placement decisions of a single subject. This was typically done when the subject believed that a specific placement target needed to be at a specific location. For example, consider someone who is trying to slow the creeps while they are inside a kill zone. Most of their slowing towers might be scattered throughout the kill zone but, in order to make sure that the creeps are not slowed too early or too late, one slowing tower is placed so that the entrance to its range is in the exact same spot as the entrance to the kill zone's range.

All three criteria are based on intent – the target might be in front of and touching an anchor but the problem solver might have only cared that the target was around and close to the anchor. Since the less precise version is how the problem solver would describe their approach to solving the problem, it is what we recorded. As with all intention-based data, values were gathered from interviews and think aloud data.

Values observed in the human data:

- **Long Before**. The target is placed sufficiently before the anchor that it does not participate in the anchor's "group", either in timing or purpose. For example, a problem solver might place several offense and slowing towers in the middle of the map to function as a single kill zone. Another set of slowing towers might be placed earlier in the map to slow a subset of the enemies, allowing the kill zone to focus on only a portion of the creeps at one time. The slowing towers are not part of the kill zone and must be placed sufficiently in front of the kill zone in order to be effective.
- **Before**. A target is placed in front of an anchor. The distance that the target is from the anchor may range from Touching to Long Before.
- **Touching Before**. The target touches the anchor but is in front of it.
- **Intersect**. The target intersects the anchor.
- **Inside**. The target is inside the anchor. Often used to place slowing towers inside the range of an offense tower.
- **Between**. The target is placed between two objects. Sometimes used as part of the Attack Window Separation strategy.
- **On**. The target is on the anchor. It is not possible to put a tower on top of another tower. Not typically used with tower ranges. Often used to place towers on corners.
- **Touching After**. Similar to Touching Before. Often used when aligning two tower's range entrances.
- **After**. The target is behind the anchor.
- **Long After**. Similar to Long Before. Often used for cleanup towers.
- **Distributed Over**. Requires a set of placement targets. The targets are placed throughout the anchor.

- **Covering All Of**. Similar to Distributed Over. The target covers the entire anchor. The most common examples are placing slowing tower ranges so that they cover all of an offensive tower's range and placing offensive tower ranges so that they cover all of the path area.

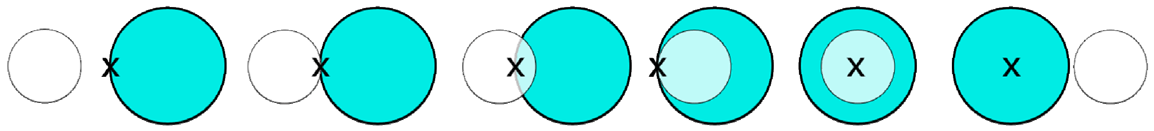


Figure 7.5. Placement Relationships. The placement relationships Before, Touching Before, Intersect, Touching After, Inside and After. The white circle is the placement target, the X is the placement anchor. In the first four examples, the anchor is the range entrance to the blue range. In the final two examples, it is the blue range.

Some values are non-directional. These include:

- **Touching**. The target is physically adjacent to the anchor. Some subjects used this to place slowing towers right next to offense towers. Also used in the FOCUSED FIRE strategy.
- **As Close As Possible**. Similar in intent to Touching but recognizing that there might not be space next to the anchor.
- **Near**. The problem solver places the target near the anchor so that they can act together but has no strong preference about distance or ordering (i.e., in front, behind, above, etc.).

Less precise relationships include:

- **Near**. Described above.
- **Near Before**. The target is mostly in front of the anchor. When applied to a tower's range, the range might intersect the anchor. When applied to a group of towers, a few might be behind the target.
- **Near After**. The target is mostly behind the anchor.

- **In Area**. The target is in the same area of the map as the anchor. A discussion of areas is given below in the Section on Map anchors.

7.4.2 Anchor

Subjects always placed their targets relative to a second object. Sometimes the object was the map (e.g., center of the map) or property of the map (e.g., a corner) but often it was other towers, properties of towers (e.g., range) or properties of properties of towers (e.g., the entry point of the range). We refer to these objects as placement anchors.

Anchors, more than most other properties, define the problem solver's approach. The problem solver must convert the map into a set of relevant features to be exploited. In human problem solvers, these features include not only obvious spatial features such as corners and corridors but innate meta-features such as path synchronization gaps and relative meta-features such as usable range and meta-features of meta-features such as attack window separation. These features, in turn, suggest strategies, in this case EXPLOIT GEOMETRY, MAXIMUM USABLE RANGE and ATTACK WINDOW SEPARATION. They also make it easy to define and execute strategies – to execute the EXPLOIT GEOMETRY strategy, one simply needs to place towers after a large path synchronization gap. In the subjects we observed, the majority of the intelligence was in identifying which features existed. Once the features were known, it was easy for subjects to experiment with them and determine which actions led to the highest scores.

There was more variation in anchors across subjects than in other properties. The types of anchors varied significantly, with concepts including spatial features and affordances (e.g., U-turns), tower groupings (e.g., kill zones), functions (e.g., number of passes), effects (e.g., group separation) and distribution (e.g., map coverage). Many anchors were used by only one or a few subjects, although subjects who used a particular anchor tended to use it on every applicable problem.

7.4.2.a Feature and Affordance

Spatial features include items such as corners, U-turns and islands. We use the term *spatial feature* to refer to a spatial arrangement that is unambiguous and context-independent. For example, we do not need to know anything about paths or creeps to know that a corner is a corner.

We use the term *spatial affordance* to refer to two separate ideas. The first refers to a spatial arrangement that functions as a specific type of spatial feature under certain circumstances. An arrangement is considered equivalent to a given feature if and only if it affords the same opportunities as the feature. For example, a U-turn has three sides and affords the opportunity for a tower placed there to attack on three sides. If the creeps do not move around all three sides (Figure 7.6, left), the position does not afford the opportunities of a U-turn. If the tower's range is not large enough to cover all three sides (Figure 7.6, right), the position does not afford the opportunities of a U-turn. If a U-turn cannot be used as a U-turn by a given tower, it is not a U-turn from the point of view of the tower and is not a spatial affordance U-turn.

It is important to note that no spatial position is ever a spatial affordance in an absolute sense. Instead, it is a spatial affordance ***in the eyes of*** another object. For example, in Figure 7.6, right, there are two towers at position B, a red tower with a large range and a green tower with a smaller range. From the red tower's point of view, position B is an island since the tower's range can reach all four sides. From the green tower's point of view, the same position is a corner because it can only reach two sides. Strategies are often tied to spatial affordances. When deciding which strategy to use, a problem solver using a red tower might use a strategy that relies on islands while a problem solver placing a green tower would not. In this way, how the space is represented affects which strategies an agent will use, which towers are used affects how the space is represented and which towers are used is determined by a strategy. From a human's perspective, there is a strong interdependence between the parts and any one part cannot be reasoned about independent of the other parts.

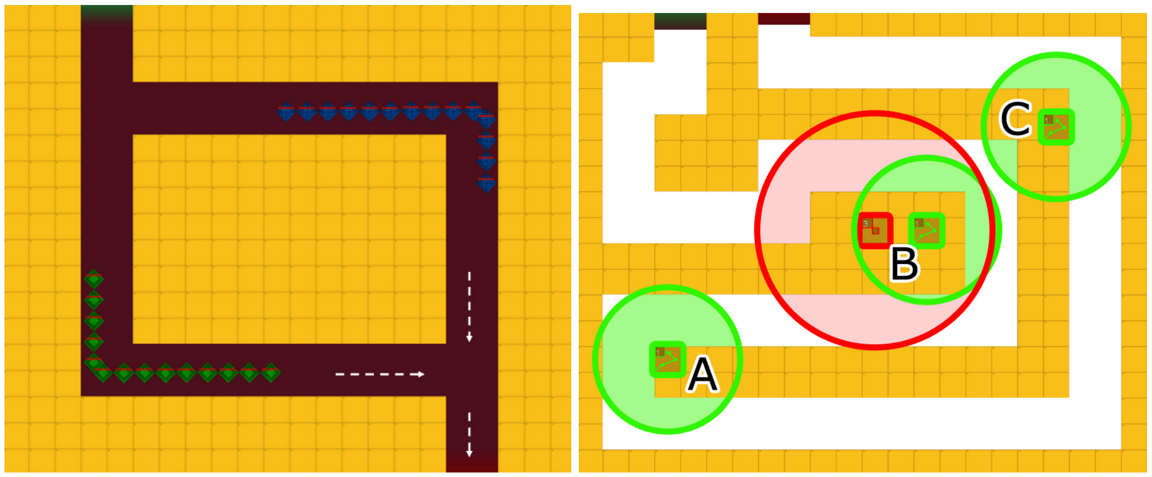


Figure 7.6. U-Turns. LEFT: Two lines of creeps move around a rectangle in the center of the map. By definition, a rectangle has four U-turns, one per side, but because no line of creeps passes three sides consecutively, the rectangle and this map has no spatial affordance U-turns. RIGHT: The towers at positions A, B and C are all on U-turns. The tower at position A can see all three sides so that position is a spatially afforded U-turn from the point of view of the green tower. The tower at position C does not have a large enough range to cover all three sides so it does not consider position B to be a U-turn. Position B isonly a corner for the green tower (right) but an island for the red tower (left).

The second use of spatial affordance refers to spatial locations that afford a given opportunity but which do not correspond to a spatial feature. For example, STRATEGY TP8: EXPLOIT GEOMETRY places towers at locations with the highest path synchronization gap (where the two lines of creeps are most out of sync). These positions allow the tower to fire more frequently but cannot be identified as a specific arrangement of geometry, nor can it be identified without context (in this case, the tower's range and the path synchronization gap). This second type of spatial affordances is defined by an ordinal property, not a specific arrangement of geometry, and is therefore covered in the section on Ordinal anchors.

The spatial features and affordances that we observed subjects using are:

- **Bottleneck**. Towers are placed around areas the subject considers to be bottlenecks. In GopherTD, creeps can move through each other and, as a result, the typical affordances of a bottleneck (slowing down the enemy) do not exist. It is not clear that there is any value to identifying bottlenecks in GopherTD or other tower defense games.
- **Corner**. A corner. From the perspective of affordance there are three types of corners. Towers on an Exterior Corner can use at least 75% of their range. Towers on a Single-Sided Interior Corner can use at most 25% of their range. Double Sided Corners are passed on both sides, changing roles from interior to exterior corner.
- **Island**. An area that is passed by a path on all sides.

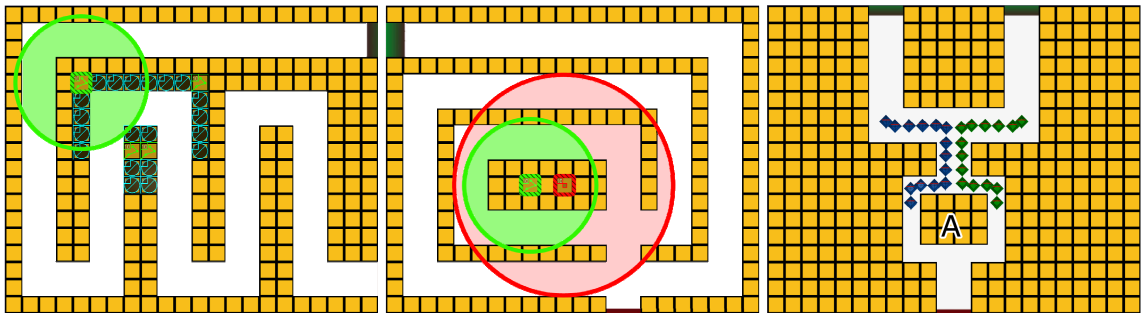


Figure 7.7. Anchors Double-Sided Corner, Island, False Island. LEFT: Double-Sided Corner (3E0400). CENTER: The center area is an island for the red tower, a U-turn for the green. RIGHT: Area A looks like an island but is a false island. It has four sides but no path goes all the way around it. The left line of creeps goes around the left side, the right line goes around the right side.

- **U-Turn**. Towers are placed on a U-turn, allowing the tower to have one long, unbroken path segment to be in range. It is possible that a few tiles in the middle of the path will not be in range (typically the interior corners) but the area is still considered a single path segment and attack window.
- **Specific Structure**. There is a specific structure the subject must find. Used when the subject has memorized a specific layout they wish to use. For subject E0100 (Figure 7.8), this is a U-turn divided by a 2x4+ wall segment two tiles tall and at

least four tiles wide. This pattern is rotation invariant but sensitive to size and path direction (i.e., the path must enter at the place considered the pattern's entrance).

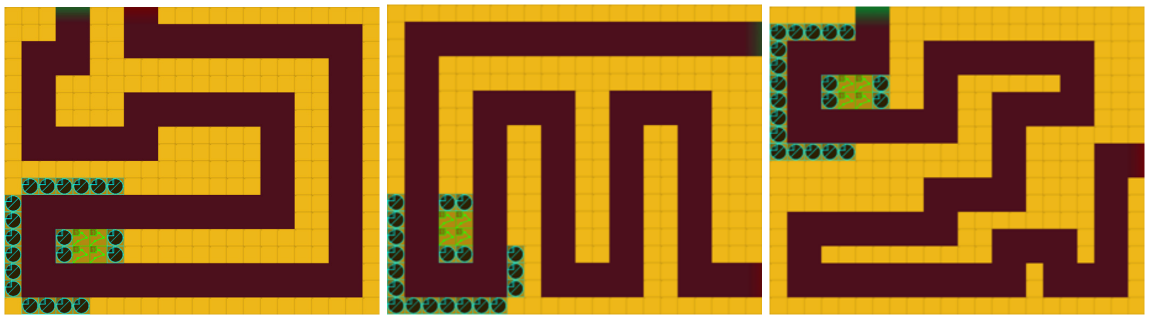


Figure 7.8. Three Solution Attempts by Subject E0100. All three look for a U-Turn around a 4x2 wall segment.

7.4.2.b Map

- **Center**. Center of the map. This sometimes, but not always, corresponds to the center of the path.
- **Region**. Some players divide maps into regions and then focus all of their energy on optimizing the placement of towers in that region. This behavior was seen in subject N0100. While some regions are obvious (e.g., the island in the map Bottleneck), many are not. What constitutes a region is determined by each subject and different subjects might identify different regions. This topic needs further study.

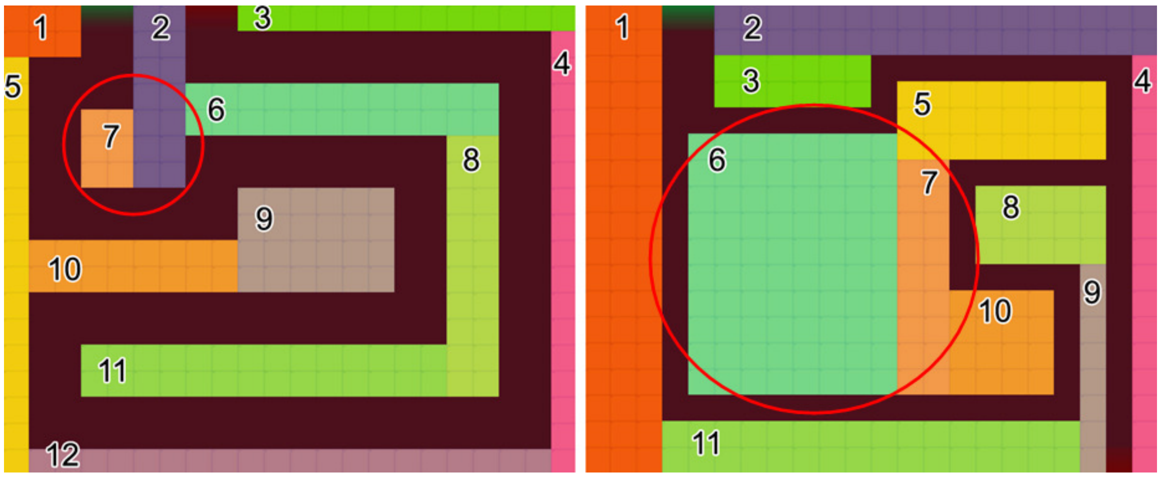


Figure 7.9. Map Area. Regions identified by the GopherTD region identification system. Areas circled represent regions that are split but which a subject might see as one region.

7.4.2.c Path

- **Path**. The anchor is the path. Not a specific part of the path (starting point, center, split point, etc.), just the path.
- **Start**. The start of the path is also the entrance to the map.
- **End**. The end of the path is also the exit to the map.
- **Center**. Center of the path. This sometimes, but not always, corresponds to the center of the map.
- **Split**. On some maps (e.g., The Void), the two paths will split and head in different directions. Splitting implies that the paths were originally together. On some maps (e.g., Bottleneck, The Frog), paths begin split, although they might later merge and possibly split again.
- **Join**. On some maps (e.g., The Void), the two paths will split, head in different directions and then join again at a later point in the map.

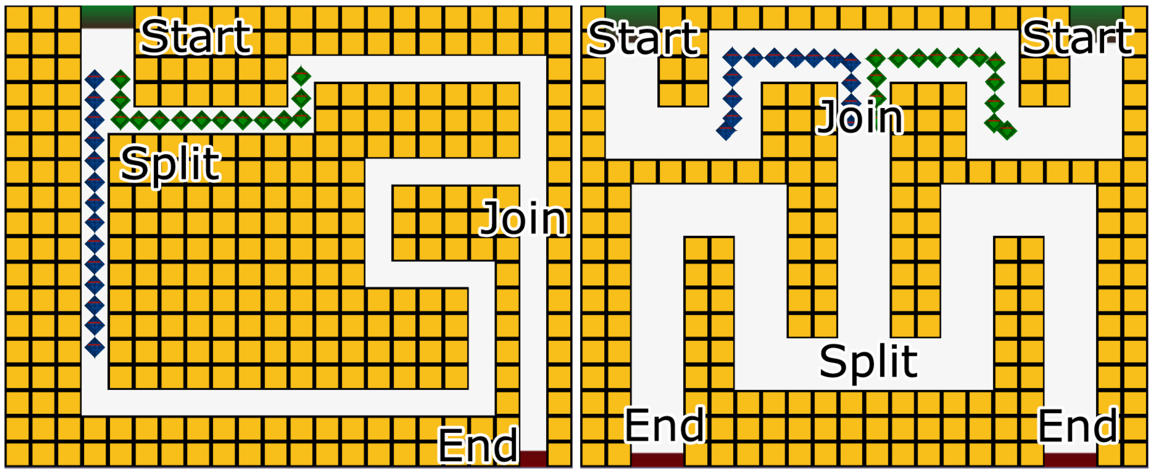


Figure 7.10. Path Positions Used as Anchors.

7.4.2.d Status / Effect

- **Group Separation**. At the beginning, the two lines of creeps are side by side. The groups can fall out of synch if one takes a longer path (e.g., they are on the outer line around a corner) or is slowed by slowing towers. Attacking one group after it has separated from the other is the basis of the strategies EXPLOIT GEOMETRY, DIFFERENTIAL SLOWING and KILLZONE JUMPING.
- **Slowed**. An area where a creep is known to be slowed. When a slowing tower slows a creep, the creep continues to move, albeit more slowly. The area that a creep travels while slow has no special visual indicator and might be outside of any tower's influence, although it will certainly be in or after the range of the tower that slowed it.

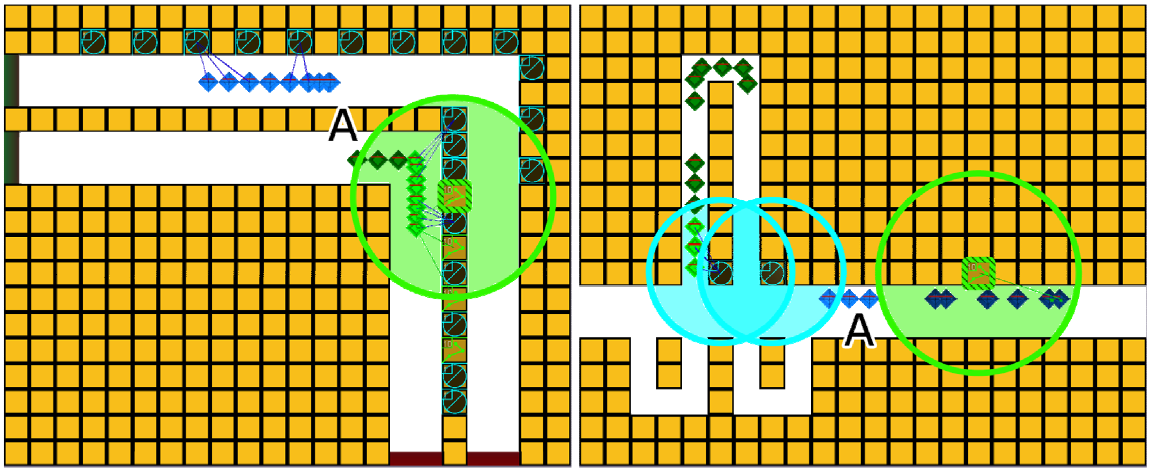


Figure 7.11. Group Separation. LEFT: Group Separation (3E0200). RIGHT: The blue slowing towers (left) slow creeps for two seconds. The creep's slowed status is still active in the area marked A.

7.4.2.e Tower and Group

- **Tower**. A tower.
- **Region**. The region a tower or set of towers is in. The concept of region is discussed in the section on Map anchors.
- **Primary Offense Group**. An area where towers are grouped with the intention of destroying healthy creeps. Also known as a kill zone.
- **Cleanup Group**. A group of towers that have the intention of destroying severely wounded creeps. Often placed near the exit of a map to prevent leakage or minimize psychological distress. See section 8.5.3 For cleanup for more information.
- **Other Group**. Other tower groups not otherwise specified. Examples include slowing towers used for the DIFFERENTIAL SLOWING strategy.
- **Chain**. Towers that are spread out in a chain rather than clustered together in a circle.

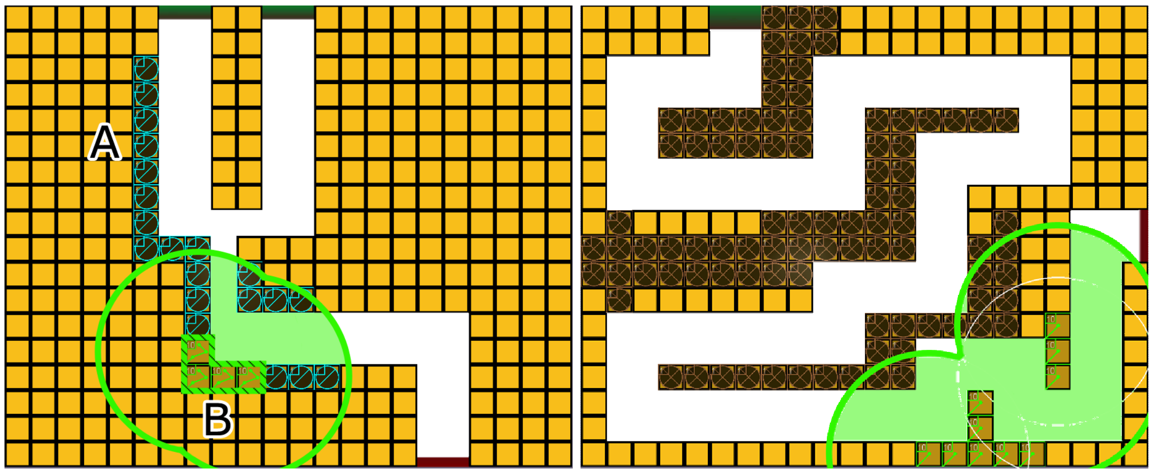


Figure 7.12. Groups as Anchors. LEFT: Primary Offense Group (3E0700). The Green3 towers at position B form an offense group. The blue slowing towers next to these towers are part of that group. The blue slowing towers at position A, however, are part of an Other Group, in this case one used for the DIFFERENTIAL SLOWING strategy. RIGHT: Cleanup Group (E0300). The subject uses brown freezing towers for most of the map but placed 10 Green1 towers near the exit to catch any creeps that made it through the primary attack area.

7.4.2.f Range

- **Range**. The range of a tower. If a tower's range covers multiple non-tangential areas, we refer to each area the range covers as an attack window. Range here refers to any or all attack windows.
- **Entrance**. The place where a path enters a tower's range.
- **Front**. The front half of the range, including the range entrance and the front half of the center.
- **Center**. The portion of the range either immediately next to the tower or between the range entrance and exit.
- **Back**. The back half of the range, including the range exit and the back half of the center.
- **Exit**. The place where a path exits a tower's range.

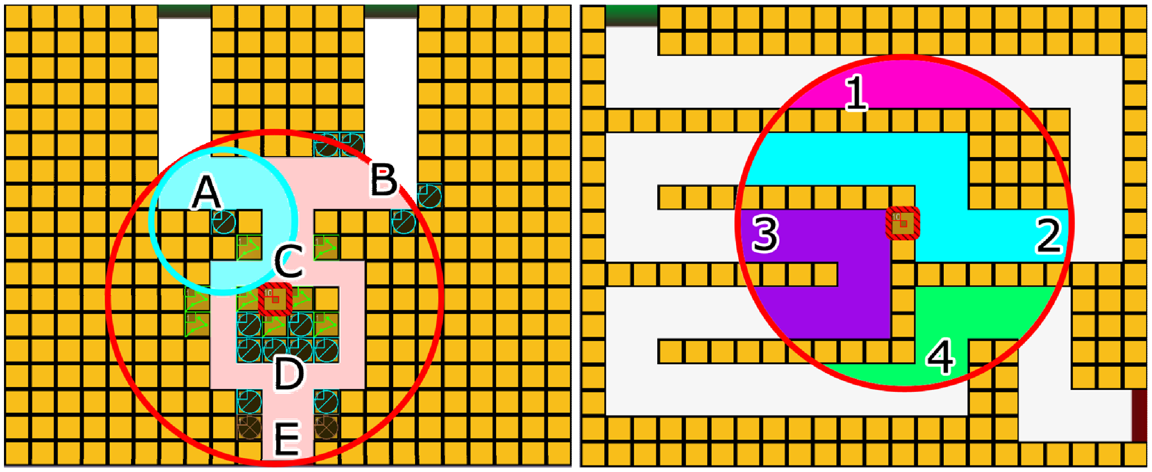


Figure 7.13. Anchors Range Entrance and Attack Window. LEFT: Range of a level 10 Red3 tower. The blue tower at position A has a range entrance that is on the red tower's range's entrance. The blue towers at position B are touching the range entrance. The eight green towers surrounding the red tower (C) are at the range center. The blue towers behind the red tower (D) are at the range back. The two brown towers at the exit of the map (E) are at the range exit. RIGHT: The level 10 Red3 tower has four attack windows at its location.

7.4.2.g Attack Window

Attack windows are an area where the tower can attack creeps. If a tower gets two chances to attack, one on each side of a wall, it has two attack windows. Specifically, this refers to a contiguous or near-contiguous spatial attack window. Attack windows are discussed more in 6.2.2 Attack Windows.

- **Attack Window**. Any attack window.
- **First**. The first attack window a line of creeps will encounter for the specified tower.
- **Middle**. An attack window that is neither first nor last.
- **Last**. The last attack window of a tower.
- **Between**. The path area between two attack windows. This is typically used by the temporal placement strategy TP6: Slow Between Attack Windows.

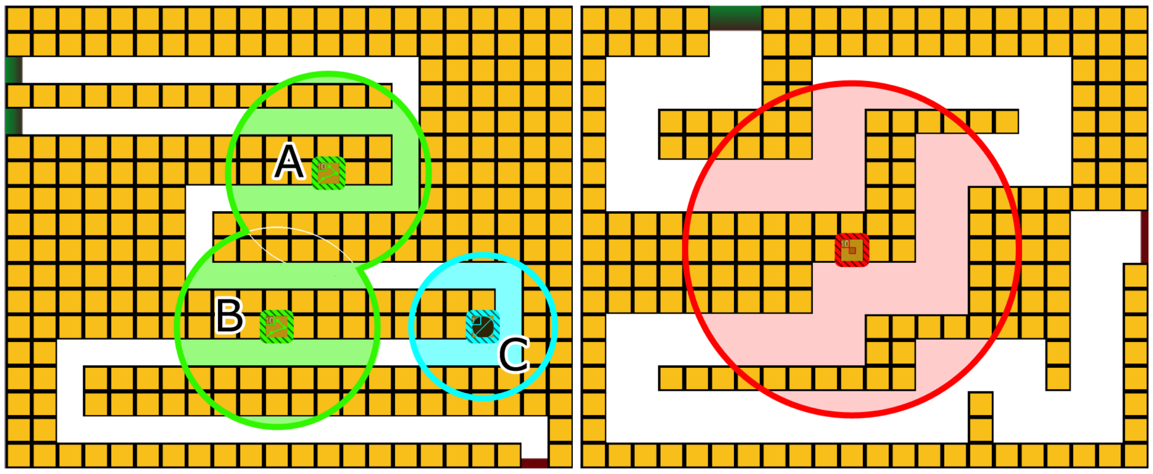


Figure 7.14. Anchors Attack Window and Number of Passes. LEFT: Tower A has more usable range than tower B. Tower B has two attack windows. Tower C, a slowing tower, is placed between tower B's attack windows in order to increase the temporal separation. RIGHT: The red tower is passed three times by the creeps.

7.4.2.h Ordinal

Ordinal anchors are features (typically meta-features) that can be measured and ranked.

- **Usable Range**. Towers are placed at the positions that give them the largest usable range size (i.e., range along path, not in a wall).
- **Number of Passes**. Towers are placed at positions that result in the creeps passing through the tower's range as many times as possible. This is equal to the number of per-creep spatial attack windows. A meaningful amount of time must separate these windows – briefly exiting the tower’s range while going around a corner should be ignored and the two windows treated as one. In GopherTD, short range towers (green) will normally only have two windows, one on each side of a wall. Long range towers (red) might have numerous windows.
- **Temporal Separation**. This is used primarily by towers with multiple attack windows. In that instance, temporal separation refers to the amount of time it takes for a group of creeps to exit one attack window and then enter the next. Temporal separation is affected by both the distance between the two points and

the time it takes to travel it (i.e., a slowed creep will take longer to move between the two points than a regular creep).

7.4.2.i Modifiers

They indicate whether the anchor is specific to a subset of creeps or a combination of multiple anchors.

- **Single Path**. The towers are intended to affect one line of creeps but not the other. An example is shown in Figure 7.12, left.
- **Pair**. The anchor refers to a pair of objects. Not often used, when it is used it typically refers to something (normally a slowing tower) placed between a pair of attack windows, towers placed between a pair of kill zones or towers placed between a kill zone and the map entrance or exit.

7.4.3 Combining Values

We defined a placement decision as a triad of `[placement target, relationship, anchor]`. In many instances, a placement decision is actually a combination of triads. In the data we saw, subjects combined criteria using prioritization, compromise or both.

7.4.3.a Prioritization

In the prioritization approach, positions are selected based on a single criterion. If more than one position matches, the tie is broken by a second criterion. This continues until there are no more criteria that the subject cares about.

In Figure 7.15, left, subject 3E0900 used a prioritized list of two criteria to place the offense towers. First, they selected the positions that would give the tower the MAXIMUM USABLE RANGE. If more than one position matched that criterion, they chose the position closest to the map exit (technically, the position where the tower's range's exit was placed right before the exit to the map).

Placement Decision:

```
1. [target=tower, relationship=maximize, anchor=usable range]
2. [target=range exit, relationship=near before, anchor=path end]
```

7.4.3.b Compromise

In the compromise approach, positions are selected based on a compromise between several criteria. If ordinal criteria are included, the criteria will likely use different units of measurement (e.g., usable range, number of passes) with different range values (e.g., usable range=0-30, number of passes=1-3). We have not collected precise data on how the compromise is achieved but based on what we saw, our initial guess is that most subjects weighted all criteria equally and for ordinal values considered only whether the values were near the top of the possible range for that map.

In Figure 7.15, right, subject N0100 initially placed their two heavy offense towers in the center island in the belief that it resulted in covering the most usable range. After some review, they decided that the spot directly across from the island was almost as good for usable range and also gave the towers three attack windows (i.e., the creeps passed through the range three times). The final spot was chosen as a compromise between usable range and number of attack windows.

Placement Decision:

```
[target=tower, relationship=maximize, anchor=usable range]
```

and

```
[target=tower, relationship=maximize, anchor=num passes]
```

7.4.3.c Combination

Prioritization and compromise are not mutually exclusive. For example, assume a problem solver who is convinced that corners are the best positions is placing a cleanup tower. The first priority would be to place the tower after the last kill zone. The second priority would be to find a good balance of corners and usable range. If there are multiple corners, they will choose the one with the most usable range, which can be accomplished by prioritizing corners over usable range. But if one position provides significantly more usable range than any of the corners, they will select the position with the most range. The only way to represent this is as a compromise between corners and usable range.

Placement Decision:

```
1. [target=range entrance, relationship=after,
    anchor=primary offense range exit]
2. [target=tower, relationship=on, anchor=corner] and
   [target=tower, relationship=maximize, anchor=usable range]
```

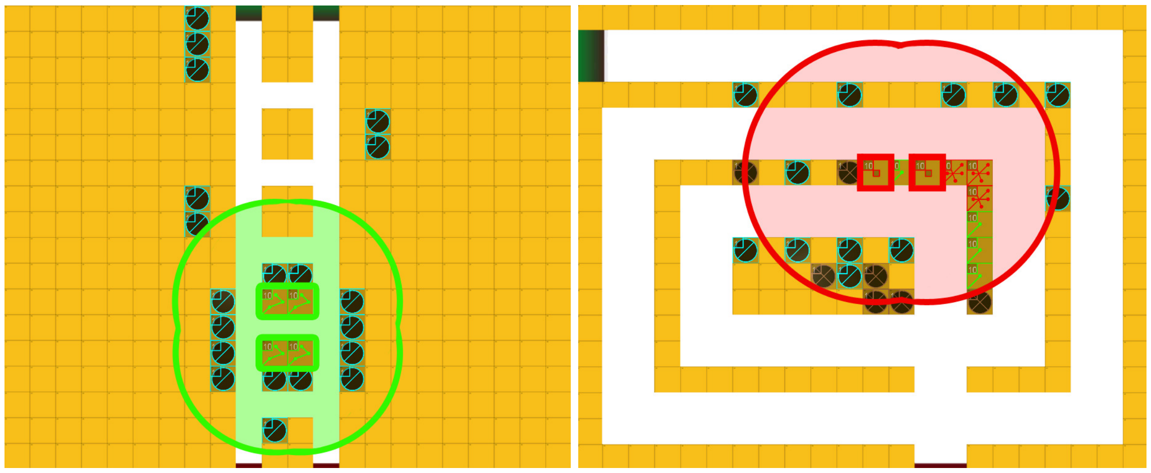


Figure 7.15. Composite Placement Decisions. LEFT: Prioritization (3E0900). The idea was to place the offense towers where they would have the largest usable range. If more than one position matched, pick the ones whose range exit is closest to the map exit. RIGHT: Compromise (N0100). The position of the kill zone (middle right corner) was chosen for its balance of centrality, usable range and number of passes.

7.5 Tower Distribution

7.5.1 Density

The density bias records whether the subject explicitly concentrates or spreads out their towers. The possible values are:

- **None**. Most subjects do not explicitly worry about density.
- **Concentrated**. Some subjects cluster their towers into a single kill zone or very small number of kill zones. All towers in all groups are placed near each other unless they do not participate in the primary offense. Excluded towers include those involved in cleanup and the DIFFERENTIAL SLOWING strategies.
- **Spread Out**. Towers are purposely distributed over the majority of the placement anchor. When the anchor is the path, this is normally paired with a "creeps always" goal such as "creeps are always taking damage" and "creeps are always being slowed". This is required for strategies such as Swamp.
- **Region**. Towers are spread out over a particular area. These areas are often columns or islands.
- **Chained**. Towers are set side-by-side in some form of chain. See Chaining for more information.
- **Touching**. All towers in a group must be as near each other as possible. This is required for strategies such as FOCUSED FIRE.

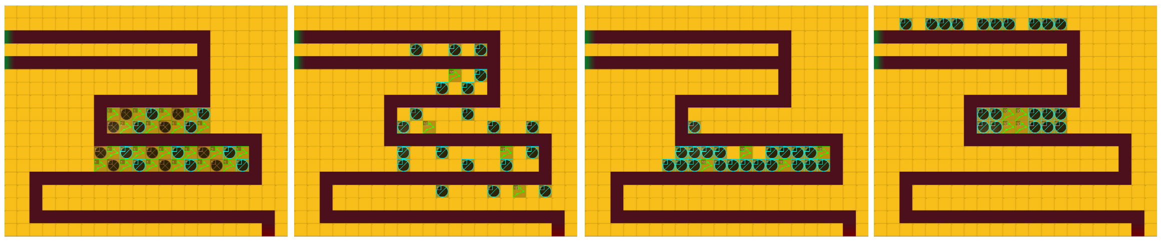


Figure 7.16. Tower Density. LEFT: Subject e2700 concentrates all their towers into one area. LEFT CENTER: Subject 3E0300 spreads out their towers. RIGHT CENTER: Subject 3E0100 places all their towers in the same region. RIGHT: Subject 3E0700 makes sure his green towers are touching. One of the blue groups forms a chain at the map entrance.

7.5.2 Chaining

There are many ways to arrange a set of towers in a chain. Chains are always assumed to follow the flow of the path.

- **None**. Towers are spread along the path but with no consistent spacing.
- **Touching**. Towers sit right next to each other, touching.
- **Spaced**. There is a fairly consistent number of spaces between each tower in the chain.
- **Distributed Over Range**. The chain fully covers a gap between two areas. Towers are spaced as needed to span the full distance.
- **Two Sided**. Towers appear on both sides of a path. This was not common.

For two sided chains, towers on each side might have a relationship:

- **None**. There is no noticeable relationship between sides.
- **Symmetric**. Each tower on one side has another tower directly across from it.
- **Staggered**. Towers on each side are offset from each other in a predictable pattern.

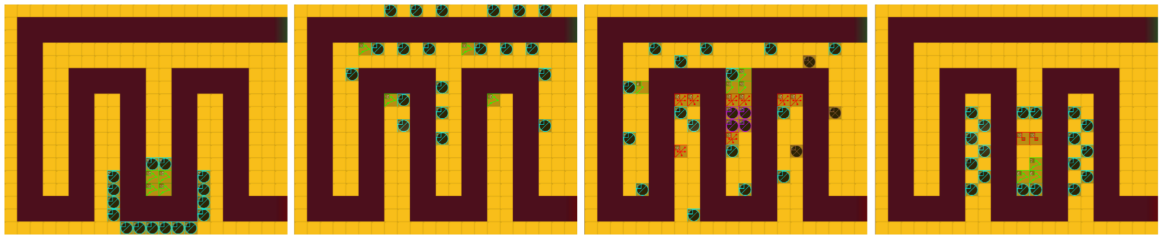


Figure 7.17. Chains. LEFT: Subject E1100's blue towers form a chain. All towers are touching except for the two interior corners. LEFT CENTER: Subject 3N0100's blue towers are spaced out. Technically, the top chain is a touching, two sided, staggered chain but the one on the third column is spaced. RIGHT CENTER: Subject N0100 equally distributes seven slowing towers across the region from the map entrance to the kill zone. RIGHT: Subject E0500 places the slowing towers in a staggered chain.

7.6 Tower Selection

In Experiment 1, subjects could choose which towers to use and how much to upgrade them. In Experiment 2, all subjects were required to use the same towers so the following values do not apply.

7.6.1 Consistency

The log files capture which towers the subjects use for each map. Using a separate tool, we analyzed how consistent subjects were in tower use. We will not go into detail about the data here and simply report the findings: which towers were used varied significantly between subjects but were relatively consistent between maps for a given subject. The type of tower was relatively consistent, varying some between the two types of slowing towers (many subjects said they weren't sure what the difference was between the blue slowing tower and brown freezing tower and so switched arbitrarily between them).

Subjects were most consistent in their use of expensive towers. It is not theorized that this is either because subjects have fewer expensive towers and thus have a smaller number to remember or because the small number of expensive towers used (never more than four fully upgraded Red3 or Green3 towers) can be subitized (i.e., they don't need to be counted) or because economics simply do not allow much variation in expensive towers (i.e., it is not possible to buy more than four fully upgraded Red3 or Green3 towers).

e0100

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD			4				20			
SP			4				20			
SB			4				20			
RtT			4				20			
E			4				20			
2i1o			4				20			
V			4				20			
BN			4				20			

e0300

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD					10	89				
SP					10	89				
SB					9	90				
RtT					5	94				
E						100				
2i1o			3			34				
V			1		16	55	9			
BN			4				20			

e0400

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD			4				4			
SP			4				4			
SB	2		2				1			
RtT			4				4			
E			4				6			
2i1o		3	3				4			
V			4	1			3			
BN			4				4			

e0500

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD	2		3				18			
SP	1		5				11			
SB	3		1				15			
RtT	3		2				15			
E	3		1				13			
2i1o			6				20			
V	1		4			8	13			
BN	3					9	14			

e0600

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD	3					16	2			
SP	1		3			11				
SB			4			11				
RtT			4			12				
E			4			12				
2i1o			4			12				
V			4			12				
BN			4			12				

e0700

	R3	R2	G3	G2	G1	B2	B1	P3	P2	P1
UnD			4				8			
SP			4			3	6			
SB			4				8			
RtT			4			2	11			
E			6				7			
2i1o			7				7			
V			5				15			
BN			5				8			

Table 7.1. Within-Subject Consistency of Tower Use. A sample of the number of towers of each type used for six subjects. Vertical axis is the map ID, horizontal axis is the tower type, table data indicates the number of each tower used on each map. Numbers in dark highlights indicate that a substantial number of towers (8+) were used.

Our initial belief was that subjects would choose different types of towers for different maps so that they could use strategies that exploited important differentiating properties of each map. Only two subjects used significantly different pieces on different maps. One (a0100) claimed that the tower selection was customized for each map, the other (E0200) said they couldn't figure out how to do well, attributed losses to the tower selection and therefore switched towers every map. There are three possible reasons for why subjects were so consistent in their choice of towers. The first is that selecting towers requires tower selection strategies. Perhaps these strategies are more advanced or come at a later stage of learning and, given more experience, the subjects would customize their tower selection to the problem. The second possible reason is that subjects have no incentive to learn how to use the other towers, either because they are doing well enough with the ones they've settled on or don't believe that using different towers on different maps makes a difference. The third possible reason is that using different towers for each map really does not help.

7.6.2 Placement Order

Every tower placed is assigned an ID equal to the number of towers that have been placed. These IDs can be used to determine the order that towers were placed. Towers that were placed and then removed do not show up in the logs, although we can tell how many towers were removed by the number of missing IDs in the log file. There is no move function in GopherTD so to move a tower, the tower must be sold and then placed in the new location. There is no way to tell from the log data how many towers were removed as a means of moving the tower rather than removed because the subject no longer wanted that tower.

While we did not conduct extensive analysis on this data, it is noted that all but one subject placed the offense towers first and the slowing towers second. This makes sense if one considers the slowing towers to be support for the offense towers.

One interesting result is that there were two maps on which the majority of subjects placed the slowing towers first. We have since classified these maps as *fast maps* (the term used by several subjects), meaning that the paths on these maps (2in1out and Bottleneck) were significantly shorter than paths on other maps. This happened both on maps subjects had experience with and maps that were novel. When asked why they placed slowing towers first, the subjects responded that time was going to be a problem on the map and they needed to deal with that first. The implication is that subjects profile the map before placing (and possibly selecting) towers, identify map-specific problems and respond accordingly. This is done even when the subject has no experience with the map.

7.6.3 Upgrade Bias

In most tower defense games, towers can be upgraded to do more damage and to have a larger range. In our experiments, all towers could be upgraded from level 1 to level 10. Each upgrade level is based on the statistics of the base (level 1) tower: cost is 50%, range 5%, power 45%. An example is given in Table 7.2.

Green1			
Level	**Cost**	**Range**	**Damage**
1	100	70	22
2	150	73	32
3	200	76	42
4	250	79	52
5	300	82	62
6	350	85	72
7	400	88	82
8	450	91	92
9	500	94	102
10	550	97	112

Red3			
Level	**Cost**	**Range**	**Damage**
1	2,500	150	30,000
2	3,750	157	43,636
3	5,000	164	57,272
4	6,250	171	70,908
5	7,500	178	84,544
6	8,750	185	98,180
7	10,000	192	111,816
8	11,250	199	125,452
9	12,500	206	139,088
10	13,750	213	152,724

Table 7.2. Impact of Tower Upgrades. Different upgrade levels for a weak and powerful tower.

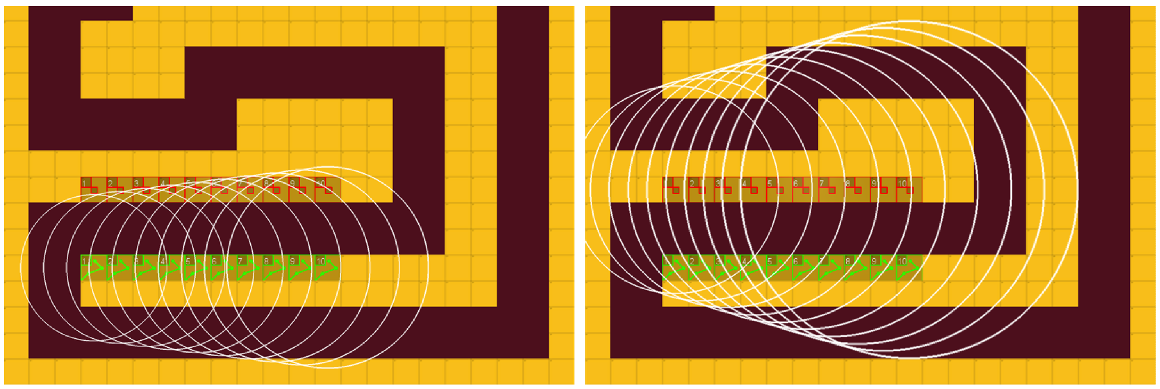


Figure 7.18. Range Sizes. LEFT: Green towers (bottom row), from levels 1 to 10. At level 7, the bottom of the range touches but does not intersect the bottom wall. RIGHT: Red towers (top row), levels 1-10. At level 5, the tower can hit the first row of creeps on the bottom row (a third corridor). At level 10, the range touches the bottom wall.

Upgrading a tower increases both range and power. There was no consensus on which trait was more valuable. The coding sheet tracks the following values:

- **None**. Subjects had access to tower and upgrade data. Several subjects spent a lot of time analyzing the data. Those that did stated that cost increased faster than power. Their strategies for offense towers focused on using large numbers of weak towers rather than upgrading. Although non-upgraded towers have less waste (excess damage caused by a shot) and are therefore more efficient, no subject mentioned this as a reason for not upgrading.

  Most subjects upgraded their heavy and medium towers. Very few upgraded their slowing towers. In some tower defense games, upgraded slow towers cause creeps to move more slowly. Most of the subjects who upgraded slowing towers said they did so because they believed this was also true in GopherTD, despite having been told at the start of the experiment that it was not.

  Upgrades were not common for cleanup towers. This makes sense as cleanup towers are intended to stop creeps that are already critically wounded.
- **Power**. In interviews, the most common reason given for upgrading power was that having more power is always good and more power always worked in other TD games.

  Two less common reasons given were support cost and building space.

  Most subjects wanted creeps to move slowly when in range of an offense tower. This requires slowing towers, so having fewer offense towers meant purchasing fewer slowing towers.

  Some maps had a few positions that were significantly better than others. If the number of towers exceeded the number of good positions, the remaining towers would have to be placed in inferior ones. Using upgraded towers maximizes the value of good positions.
  - **High**. When subjects upgraded towers, they tended to fully upgrade them. Powerful towers kill creeps faster. For some subjects, fully upgrading a

    tower's power is done for some of the same reasons as the FOCUSED FIRE strategy.
  - **Medium**. Very few subjects upgraded towers to levels 4-7. It appears that, if a tower is worth upgrading, it is worth upgrading fully.
  - **Minor**. This was an uncommon choice.
- **Range**. The larger the range, the more time the creeps spent being attacked.
  - **High**. The most common choice for range upgrades. If the range was increased enough, the tower was able to cover another corridor, giving the tower an additional attack window.
  - **Medium**. This was an uncommon choice.
  - **Minor**. This was an uncommon choice.
  - **Maximum Usable Range**. Several subjects noticed that, past a certain level, additional range did not help as it went past the path and into the wall (Figure 7.19), a problem several subjects referred to as "ranging into the wall". Subjects who noticed this upgraded their towers so that they maximized the amount of path area in range (maximized usable range) without going into the walls.
- **Both**. After the tests, subjects were asked why they upgraded. Most subjects said they were trying to get more power or range, but a small number said they had been considering both. Although this seems logical, subjects in post-test interviews seemed susceptible to suggestion and often stated what they thought they should have done rather than what they actually did so there is no way to verify this answer.

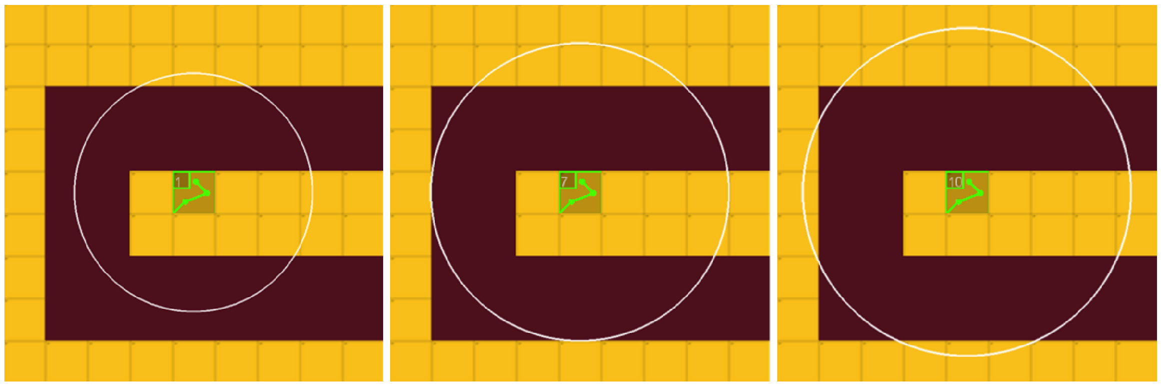


Figure 7.19. Relationship of Range to Usable Range. LEFT: Green tower level 1. CENTER: Green tower level 7. Range touches the left and bottom walls. RIGHT: Green tower level 10. Range extends into the left and bottom walls, where it is not useful. In all three cases, the top of the range circle extends into the wall. In all three cases, the tower has been moved back one square to get additional usable range, a common practice.

In addition to avoid ranging into walls, there were two other reasons given for adjusting range or power.

First, if the subject had money but not enough to purchase another heavy, they might use the money to upgrade an existing tower.

Second, range was sometimes decreased to avoid covering an additional corridor. This was done to control targeting – a tower intended to cover an exit might ignore creeps in that area if it was focused on creeps in another, less important area. This could also be accomplished by moving towers so that they covered less area.

7.6.4 Slowing Bias

There are two types of slowing towers. Blue towers slow creeps, brown towers stop them. Blue towers are cheaper (300 vs. 500) and affect multiple creeps. Both fire every two seconds and their effects last for two seconds.

There was no consensus as to which tower was best. Many subjects said they tried both but couldn't see any significant difference.

Values tracked were:

- **Slow**. The subject primarily used blue towers. This was the most popular choice. Because blue towers slowed four times as many creeps as brown, many subjects considered blue the more economical choice.
- **Frozen**. The subject primarily used brown towers. This was a common choice but not as common as selecting the all-blue approach. It was more popular for cleanup towers. This was the only choice for people using the Swamp strategy.
- **Mostly Slow**. Most towers were blue but between a quarter and a third of towers were brown. Subjects who did this said they couldn't tell the difference between the two types of slowing towers but knew blue was cheaper.
- **Mostly Frozen**. Most towers were brown but between a quarter and a third of towers were blue. This was not a popular choice.
- **Mix**. The subject used a roughly equal number of blue and brown towers. Most subjects who did this said they couldn't tell the difference between the two types.

7.6.5 Cleanup Strategy

Heavy and medium towers are designed to stop strong, healthy creeps. Most subjects spend the majority of their money on these towers. The intent is usually for these towers to stop every creep but sometimes a few creeps will make it past these towers. This is called leakage. If the creeps are still healthy, there is likely little that can be done – the core of the solution attempt was a failure. If the creeps are severely wounded, only a little force is needed to stop them. Some subjects used dedicated towers to deal with these.

There are several ways to execute a cleanup strategy. As the goal of the cleanup tower is to destroy weakened creeps, at least one offense tower is needed. The simplest option is to use a dedicated offense tower. Because the creeps are weakened, these towers are typically much weaker than the heavy towers. Because leakage is more likely to involve several weakened creeps rather than one healthy one, cleanup offense towers are often (but not always) fast, weak towers rather than strong, slow ones.

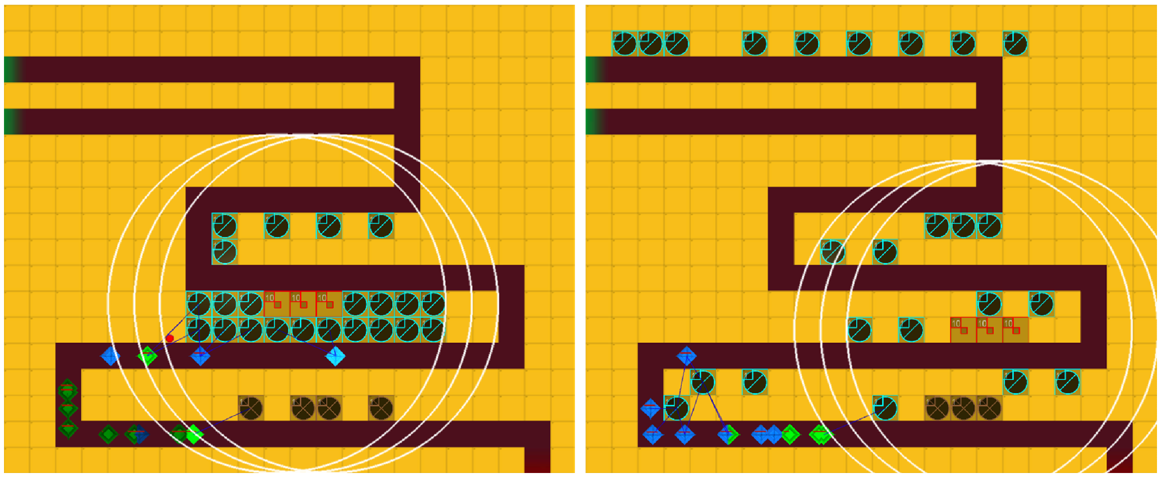


Figure 7.20. Re-Use Heavy as Cleanup. LEFT: A contrived example showing the wrong way to re-use a heavy as cleanup. The heavy towers are still dealing with creeps in the primary attack area when creeps arrive in the cleanup area. The tower is being asked to play roles of heavy and cleanup concurrently. Since the creeps in the primary attack area are closer, the towers will ignore creeps in the cleanup area, allowing them to escape. RIGHT: a0200 re-uses their heavy towers as cleanup towers. All creeps leave the primary attack area before the first enters the cleanup area, giving the tower time to switch roles.

A second option is to use a strong tower as both a heavy and cleanup. Typically, a tower with a large range is used (in GopherTD, an upgraded red). The tower is placed so that it covers a desirable position as a heavy tower (the primary attack area) but also where part of its range will cover the path near the map exit (the cleanup area). The advantage of this approach is that a powerful tower can be used for cleanup and no money needs to be spent. Ideally, creeps enter the primary attack area, the tower fulfills the role of a heavy, all of the creeps leave the attack area and then the first leaked creeps enter the cleanup area and the tower fulfills the cleanup role. The dangers of this approach are that the tower will either fail to select the best target (typically the one closest to the map exit) or be unable to play both roles. If creeps are still in the primary attack area when the first creeps enter the cleanup area, either because the two areas are too close together or because too much slowing was used in the primary attack area, the tower will focus on the closest creeps (typically those in the primary attack area) and not switch to cleanup (Figure 7.20).

Using cleanup towers was uncommon in phase 3 as subjects were only given four offense towers, making dedicating one to cleanup an expensive proposition. The towers did not have a large range, making re-use of towers difficult.

The values tracked included type of slowing and offense towers. Multiple selections were allowed.

- **Slow**. The subject used blue slowing towers to cause leaked creeps to move slowly through the cleanup area.
- **Freeze**. The subject used brown freezing towers to completely stop creeps in the cleanup area. This is a common option for cleanup areas. Outside of cleanup areas, blue towers are more common because they affect multiple targets. Normally, only a few creeps would be leaked, so this advantage is less important in cleanup areas. In contrast, given that the cleanup area is the last place a creep can be stopped, brown's ability to completely stop a tower for two seconds is more valuable.
- **Light Offense**. The subject used very weak offense towers. In GopherTD, this corresponds to the extremely cheap Green1 towers (phase 1 and 2 only).
- **Offense**. The subject used a normal powered offense tower. Towers are normally weaker than heavies (otherwise they would be another heavy).
- **Re-Use Heavy**. A heavy tower is also used as a cleanup tower as explained above.

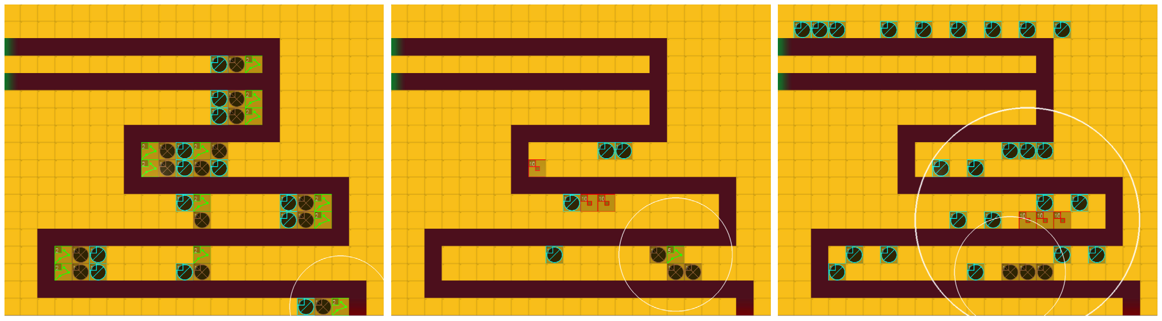


Figure 7.21. Cleanup Groups. LEFT: e2800 places cleanup towers at the map exit. The towers (one offense, two slowing towers) are placed below the path rather than above to avoid covering the above corridor, guaranteeing that the cleanup towers focus solely on creeps about to escape. CENTER: a0100's primary attack towers are slow, powerful red level 10 (center of map) supported by blue slowing towers while the cleanup towers are a fast, weak green level 5 supported by brown freezing towers. RIGHT: a0200 re-uses their heavy towers as cleanup towers. They have no dedicated offense cleanup towers, only three freezing towers (all non-cleanup slowing towers are blue). The freezing towers are placed at the range entrance to the heavies' last attack window.

7.7 Representational Complexity

7.7.1 Size of State Space

All maps used in the experiments were 22 x 18 grids (396 tiles). Towers may only be placed on walls. The number of wall tiles ranges from 86-350, with an average of 254 (64% of space).

There are 11 towers, each of which can be set to one of 10 power levels. Each tower has a cost ranging from 100–13,750. Subjects were given a budget of 50,000. If the subject spent all their money, the number of towers to place ranges from 3-500.

Assuming a map with 254 wall tiles, there are $\sum_{i=1}^{254} \binom{254}{i} = 3\times 10^{76}$ ways to place an arbitrary number of level 1 Green1 towers, the cheapest tower available. There are approximately 10^{78} ways to place an arbitrary number of towers of a single type and level. Because the type and level of towers can be mixed, the actual number of possible tower arrangements is much higher.

7.7.2 Size of Solution Description

In our experiments, subjects were presented with a map and asked to place a set of towers where they believed they'd be most effective. This placement is a proposed solution to the map. Each of these proposed solutions was captured on a coding sheet.

The placement of each tower is described by a Placement Decision consisting of a placement target, anchor and relationship. There are 7 targets, 19 relationships and 37 anchor values. An anchor can have more than one value, with each pair of values having one of two relationships, precedence and compromise (7.4.3 Combining Values). In theory, one could use every anchor value in a single placement decision. There are $\sum_{i=1}^{37}(2\times i)! = 3\times 10^{107}$ unique combinations of criteria for an anchor. Removing mutually exclusive values lowers that number slightly. In practice, the number of anchor values used in a single placement decision is 1-4, lowering the number of unique anchors to $\sum_{i=1}^{4}\binom{36}{i} = 83{,}538{,}960$. The number of unique placement decisions is 7 * 19 * 83,538,960 ≈ 11 billion.

As mentioned in the previous section, subjects can purchase up to 500 towers, although, on average, there is only room to place 254 and in practice subjects use significantly less. Towers belong to functional groups. Groups can be as small as one tower, although in our experiments this was somewhat unusual for towers in the Heavy role and very unusual for towers in the Slowing role. Placement decisions are recorded by functional groups.

7.8 Representational Limitations

Ordering Constraints. Subject a0200 slows all creeps as soon as they enter the map and then maintains that slowness until the creeps reach the offensive towers. This is accomplished by placing a chain of three blue slowing towers at the entrance and a series of single blue slowing towers from there to the start of the range of the core offensive towers. The first group can be coded as having an anchor Map:Entrance and the second group as having an anchor Range Entrance:Long Before. However, there is no way to capture that the second group starts after the first group ends.

Weighting. Using the current coding sheet, it is possible to record that a subject's decision was based on two criteria (e.g., usable range size and center of map). It is assumed that the criteria are of roughly equal importance. There is currently no way to say that one criterion was three times more important than the other (e.g., 75% usable range size, 25% centered). Likewise, when recording the slowing preferred effect, the coding sheet can record mostly slowing towers, mostly freezing towers or mix but cannot track specific ratios.

Detailed Placement Information. Subject E0100 looks for U-turns where the dividing wall is exactly two tiles tall and at least four wide (assuming the U-turn is horizontal). The first two tiles are blue slowing towers, the next two are green offensive towers. The green towers can be coded as having an anchor of Feature:U-Turn but there is no way to capture that the green towers are placed exactly one tile away from the end of the U-turn.

Temporal-Based Positioning. Subject A0200 slows all creeps as soon as they enter the map and then maintains that slowness until the creeps reach the offensive towers. To accomplish the maintenance goal with the fewest possible resources, towers are placed at positions where the start of the tower's range is at the place where the creeps move from slowed to normal speed. This position is determined by estimating how far a slowed creep can move in two seconds (the duration of the slowing effect). There is currently no way to capture this.

Goals, Strategies and Levels of Abstraction. Subject A0200 wishes to win the game.

1. Their strategy for doing this is to maximize the attack time of his offense towers.
2. Their strategy for doing this is to split the enemies into two groups and then attack them separately.
3. Their strategy for doing this is to slow down one group so that their offensive towers have the maximum amount of time available to attack the second group and then, once that time is elapsed, have the maximum amount of time available to attack the first group.
4. Their strategy for doing this is, when it's possible and valuable, to place the slowing towers so that they can only reach one group, place enough slowing towers at the entrance of the map to slow all members of that group and then place maintenance slowing towers along the path so that they re-slow the group members once the earlier slowing effects have worn off.
5. Their strategy for doing this is, when it's possible and valuable, to place a chain of three Blue1 towers such that the first tower's range begins at the map entrance and then place Blue1 towers every other square up to the start of the offensive towers' range or, if they don't have sufficient resources to do that after resources have been allocated to other goals, to place Blue1 towers every other tile after the initial chain until they run out of resources.
6. Their strategy for doing this on the problem Switchback was to place three Blue1s one tile away from the map entrance for the rightmost group, place a Blue1 two tiles after that (still one tile removed from the path), place another Blue1 two tiles after that and then place four Blue1s every other tile after that.

Using the coding sheet, we can capture the facts mentioned in level six but not the goals and strategies that caused that solution to be generated nor can we capture any of the more abstract facts, goals and strategies in levels one through five.

Occasion Setters and/or Affordances. Subject A0200's strategy is to, when it's possible and valuable, use slowing towers to separate the two groups of creeps, allowing the offensive towers to attack the groups one at a time. This is used when slowing towers can be placed such that they only affect one group and when the two groups can be fully separated before they enter the range of the first offensive tower. Groups can be slowed differentially when the creeps use separate entrances and when the wall is thick enough that slowing towers can be placed one tile away from the path. Groups can be fully separated before entering the offensive towers' range if the average path length of two side-by-side paths to the offensive towers is at least a certain minimum distance. This distance can be shorter if the map geometry contributes to the separation, which happens when paths have different lengths or pass by one or more non-compensatory corners (i.e., corners whose separation effects are not undone by a subsequent corner). Using the coding sheet, we can capture whether the subject used a certain strategy but not what features and reasoning process were used to determine whether the strategy was relevant.

Variation, Noise, Learning and Refinement. While our experiment protocol is designed to result in subjects maintaining stable strategies in testing, there is variance. This comes in many forms. Subject A0200 placed one slowing tower in an unexpected place, disrupting a chain, because he wasn't paying attention. Subject E0100 ringed the outer walls of a horseshoe with slowing towers but after a few maps refined this slightly by not placing towers in the corners as they only used 25% of their range rather than 50% for the others. Near the end of testing subject N0100 learned a new strategy that resulted in qualitatively different solutions for the last two problems. The coding sheet captures strategies used on each problem but does not capture whether the strategy has changed from earlier problems, either because of noise, incremental refinement or significant insight.

Chapter 8

Strategies

What follows is a list of the strategies observed in subjects in Experiments 1-3 of the GopherTD experiments. The list includes three types of strategies: primary, satisficing and other. Primary strategies represent key ideas used by subjects, either explicitly or implicitly.

Satisficing strategies include simplifications of primary strategies. For example, most subjects used some variation of the MAXIMUM USABLE RANGE strategy, which says to place the towers where their range area covers the largest amount of the enemy path. Some subjects decide that corners (u-turns, etc.) cover the most usable range and so, rather than looking at all possible positions on a map, they only look at corners. While towers on corners might often cover the most usable range, it is not guaranteed. This type of strategy saves the subject time and memory but potentially decreases their accuracy.

The "other" strategy category covers strategies that do not belong in the former two categories. One example is the MAXIMUM USABLE RANGE COMPROMISE strategy which tries to balance MAXIMUM USABLE RANGE with a competing strategy such as MAXIMUM NUMBER OF PASSES.

8.1 Spatial Placement Strategies

Spatial placement strategies use only spatial properties to determine where to place towers. There are two primary types of spatial strategies – maximize the amount of area covered and maximize the number of attack windows.

8.1.1 To Increase Coverage Area

The primary goal of a tower defense puzzle is to prevent the creeps from reaching the map exit. To do this, creeps must be destroyed. Defense towers do a fixed amount of damage per shot, shoot at a constant rate and never miss. The only variable that can be manipulated is how often the tower fires, known as the tower's *active time*. Towers only fire when there is a creep in their range circle. By controlling the amount of time one or more creeps spends inside a tower's range circle, one can control the tower's active time.

If a tower's range circle covers more of a path, the creeps spend more time in the tower's attack range, all other things being equal. For this reason, increasing the area covered increases the tower's active time proportionally. Space (area) is used as a substitute for time (active time).

Strategy SP1: Maximum Range - Physical Space

Maximize the size of a tower's range. This is done in two ways: upgrade towers (upgraded towers have higher ranges) and avoid exterior walls (where part of the range will be off the map). This strategy focuses on absolute range, not usable range. Usable range is discussed in the next strategy.

Strategy SP2: Maximum Usable Range - Physical Space

Maximize the size of a tower's usable range. *Usable range* is the amount of space in a tower's range that a creep will move over. Path cells are usable range, wall cells are not. There is no advantage to having a range that covers mostly wall cells as it does not increase the amount of time creeps spend in a tower's range and thus does not increase the tower's active time.

This strategy focuses on physical path cells. These are the path cells that can be seen on the map. They are not a direct measure of how much time creeps spend in range, although on many maps it is an accurate measure.

Satisficing Strategy: Color Ratio

One subject determined which position was best by placing the tower's range circle over the locations of interest (in their case, corners, u-turns and "*skinny walls*") and measuring the amount of maroon (floor tiles) vs. the amount of gold (wall tiles). The spots with the highest percentage of maroon were selected.

In concept, this is a measure of usable range. It is, unfortunately, a flawed one. It measures physical space, not aggregate traffic volume, and thus suffers from the same disadvantages of spatial measurements as the strategy MAXIMUM USABLE RANGE – PHYSICAL SPACE. In addition, this strategy conflates floor space with path space. There are three types of tiles in GopherTD – walls (gold) and the two types of path tiles, path (maroon) and empty (also maroon). While most floor tiles are path tiles (i.e., space that creeps will cross), some are not. Maximizing the amount of maroon space covered by a tower is an easy strategy for a human to implement given human visual capabilities but it results in covering empty floor space that is no more valuable than the gold walls the strategy seeks to avoid.

Strategy SP3: Maximum Usable Range – Traffic Volume

Maximize the size of a tower's usable range as measured by the total amount of time that normal (non-slowed) creeps are in range. This varies from physical space in two ways: multiple creeps can be on the same physical cell (and multiple lines of creeps can share the same physical cells) and creeps can pass over the same physical cell more than once. On maps where lines of creeps do not merge into the same set of cells and creeps do not cross the same cell more than once, physical space and traffic volume are equivalent.

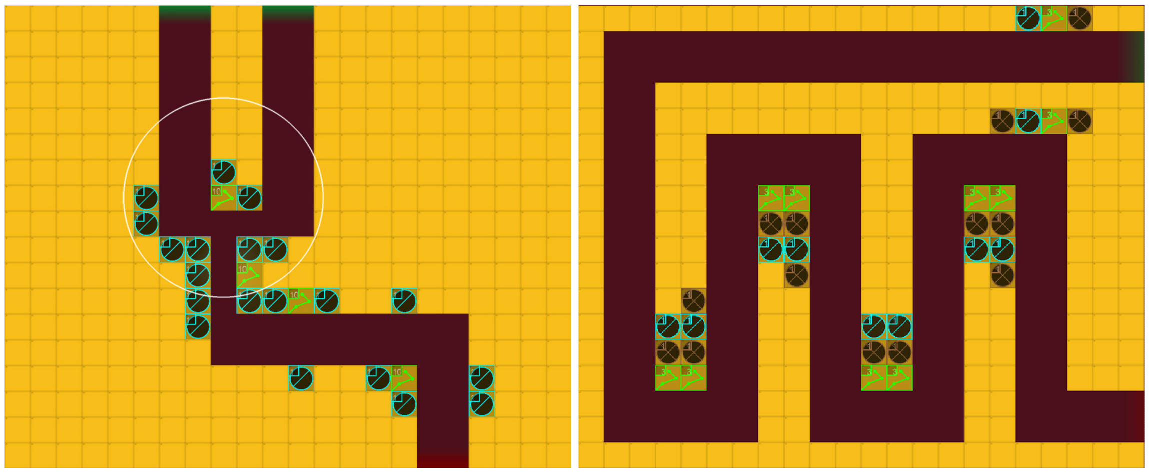


Figure 8.1. Strategies Maximum Usable Range-Physical Space and Use U-Turns. LEFT: MAXIMUM USABLE RANGE – PHYSICAL SPACE (3N0500). The majority of the area covered by the range of the tower at the bottom of the column where the two lines merge is path area. Unfortunately, because only one line of creeps runs through each side of the range, the amount of time the tower will fire is the same as if the tower had been placed on the bottom horizontal area, where half the range would be in the wall and therefore unusable. RIGHT: USE U-TURNS (e2800). The subject places towers in U-Turns.

Satisficing Strategy: Use Corners

Many subjects felt that corners gave the MAXIMUM USABLE RANGE (either physical space or traffic volume). When determining where to put their primary attack towers, they only looked at positions that were exterior corners (i.e., where a path went around 270° rather than 90°). These positions might or might not have the MAXIMUM USABLE RANGE.

Satisficing Strategy: Use U-Turns

Many subjects felt that U-turns gave the MAXIMUM USABLE RANGE. Most, though not all, GopherTD maps had U-turns. When determining where to put their primary attack towers, they only looked at positions that were U-turns. These positions might or might not have the MAXIMUM USABLE RANGE.

It is important to note that a U-turn is a spatial affordance, not a spatial feature. Whether or not an area is a U-turn depends on both the tower placed there and the flow of the path around it. If an area is too wide for the tower's range to reach all three sides, it is not a U-turn. If no line of creeps moves around all three sides, it is not a U-turn. GopherTD maps have significantly more U-turns as spatial features than as useable spatial affordances.

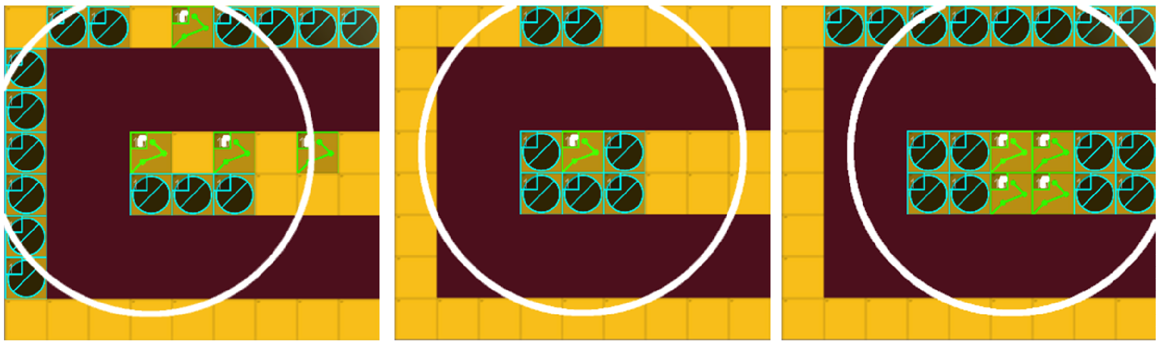


Figure 8.2. Strategy One Back for Range. LEFT: Subject 3E0500 places a level 10 Green3 offense tower. The left end of the range covers a column of the wall. Usable range=27. CENTER: ONE BACK FOR RANGE (3N0500). The offense tower is moved back from the U-turn by one tile so that its range is not wasted in the wall. Usable range=30. RIGHT: Subject 3E0700 places the offense tower two tiles away from the end of the U-turn. Usable range=28.

Satisficing Strategy: One Back for Range

Towers have a given range, with upgraded towers having a larger range. In phase three, all subjects used level 10 Green3 towers. When placed on the end of a U-turn, the range of this tower covers the path area next to the U-turn plus a large part of the wall beyond it. The One Back for Range strategy says to never place towers at the end of a U-turn, always back them up. This particular strategy is specific to GopherTD and level 10

Green3 towers but the nature of the strategy is universal – once a particular observation has been made (in this case, that the range goes into the wall) and a rule has been learned (in this case, move towers back by one), it can be used directly across all problems in the same category without resorting to a more general but significantly more memory- and time-intensive strategy (in this case, achieving MAXIMUM USABLE RANGE by measuring all options and taking the best).

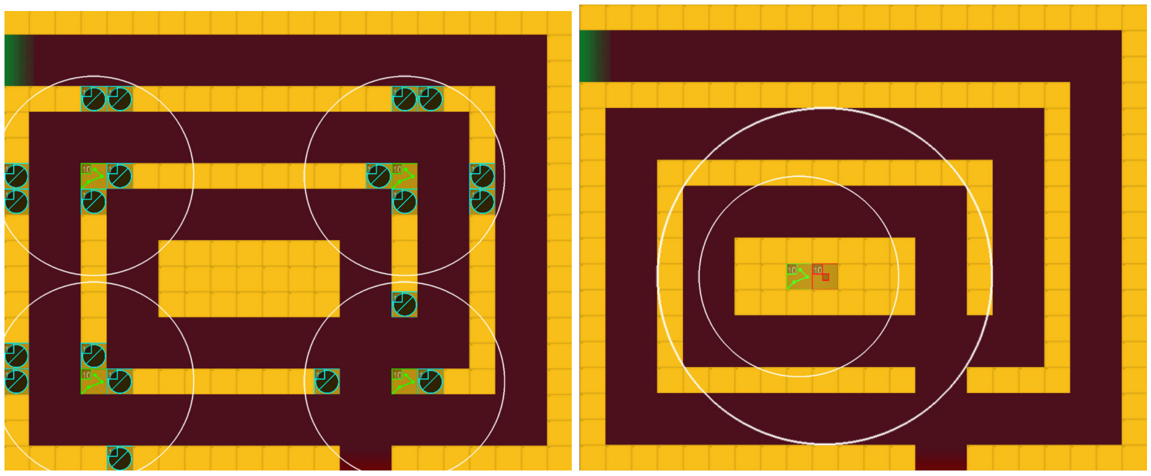


Figure 8.3. Maximum Usable Range – Corner (3N0700). LEFT: Offensive towers are placed on corners. RIGHT: MAXIMUM USABLE RANGE – Island. Islands cover four path segments, as opposed to the three covered by U-Turns.

Satisficing Strategy: Use Islands

Many subjects felt that islands (a small to medium rectangle that the creeps passed on all four sides) gave the MAXIMUM USABLE RANGE. Only a few maps in GopherTD had islands so this was unlikely to be a subject's only strategy. More likely the subject used this as the first strategy and if it didn't match they fell back to a similar strategy such as Satisficing Strategy: Use U-Turns.

Note that when subjects talk about islands, like U-turns, they are normally talking about the affordance, not feature. Whether an area is an island depends on whether the range of the tower placed there covers all four sides. If it can only cover three sides, the area is effectively a U-turn. The same area might be an island, U-turn, corner or wall depending on the position inside the area and the range of the tower (which in turn depends on the type and upgrade level of the tower).

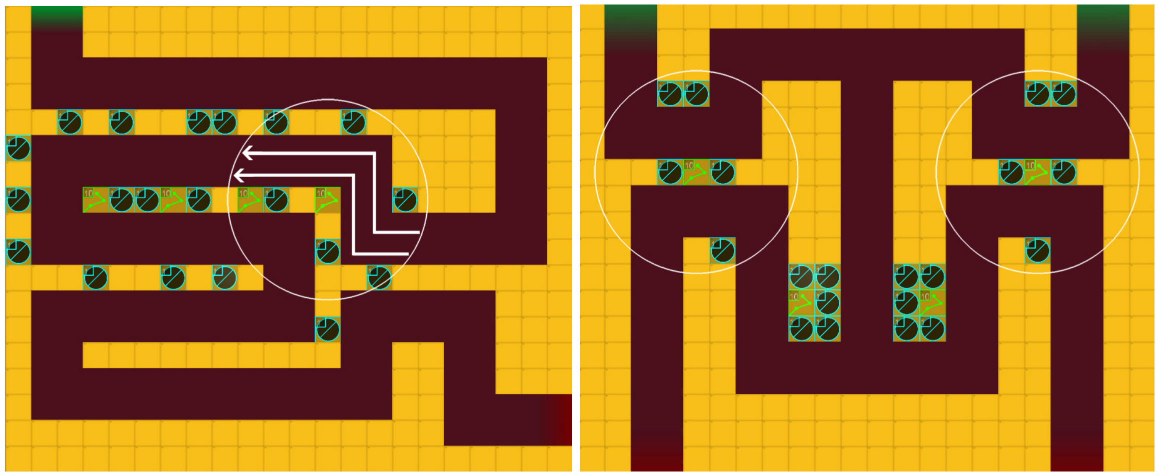


Figure 8.4. Maximum Usable Range – Zag and Skinny Wall. LEFT: MAXIMUM USABLE RANGE – ZAG (3N0900). A zag covers three path segments, like a U-Turn, but two segments (here, the horizontal) travel in the same direction. Unlike a U-Turn, the path can come back for a second pass (interior corner above). RIGHT: MAXIMUM USABLE RANGE – SKINNY WALL (3N0700). A wall that is only one tile thick has more path area (maroon) in the tower's range than wall (gold).

Satisficing Strategy: Use Zags

Subject 3N0900 focused on "*zags*", Z-shaped areas where there are three path segments but only two directions (see **Figure 8.4**), as opposed to U-turns, which have three path segments traveling in three different directions. Both result in a single continuous attack on a line of creeps. Zags appear on many of the maps that do not have U-turns. Depending on the specific layout, zags can cover as much path area as a U-turn. Positions on a zag, however, might be passed twice, which does not happen with U-Turns. Only one of the 58 subjects studied mentioned zags.

Satisficing Strategy: Use Skinny Walls

One subject looked primarily at corners, u-turns and "*skinny walls*". A skinny wall is one which is long in one direction and only one tile wide in the other. The reasoning behind choosing skinny walls is that it had more "maroon" (floor tiles) in the tower's range. The importance of this is explained in Color Ratio, the strategy they used in combination with this one.

Strategy SP4: Maximum Percent of Path

Several subjects (primarily domain novices) distributed their offensive towers across the map. This can be interpreted in several ways. Some said they were trying to cover the entire area that the creeps moved through. Some said they wanted to make sure creeps were always being attacked. Some said they were attempting to minimize the amount of overlap between the ranges of offensive towers. Some said they believed that tower damage did not stack (i.e., if a creep was taking damage from one tower, it could not take damage from another tower; this is not true for offensive towers but is for slowing towers – a creep that is already slowed cannot be slowed more). It is not clear that this strategy has any value, although we have not explicitly tested it. This strategy was not explicitly stated by any domain experienced subject, although several did cover the majority of the path area (notably those who use SWAMP ATTACKs, in which towers are spread out but for very different reasons).

8.1.2 To Increase Chances to Attack

Strategy SP5: Maximum Number of Passes

Many of the walls in GopherTD are passed by the creeps twice, once on each side. An exception to this are the exterior walls (the ones that form the boundary of the map), which are (normally) only passed once. In some cases (e.g., where the path crosses itself) a spot might be passed three or more times. Each time a line of creeps passes is another opportunity for the tower to attack. All other things being equal, a tower that is passed twice is active for twice as long and is therefore twice as effective.

The passes referred to here are temporal, not spatial. The creeps must pass through the tower's range one time, leave and then come back into range later (the second pass). For this reason, U-turns are not included in this strategy. Creeps pass by three sides of a U-turn but do so in the same attack (time) window. Islands are excluded for the same reason.

Satisficing Strategy: Two Passes

Some subjects wanted to maximize the number of passes the creeps made past the tower but stopped when they found a position that creeps would pass by twice. This resulted in subjects missing superior positions (in terms of number of passes). For most maps, two is the maximum number of passes, which probably explains why subjects did not look for better areas.

8.1.3 Compromise Strategy

A compromise strategy combines two or more competing, potentially mutually exclusive strategies into one strategy. Often one of these strategies is one of the MAXIMUM RANGE strategies. An example is combining the strategies STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME and STRATEGY SP5: MAXIMUM NUMBER OF PASSES. The spot that gives the maximum usable range might have a usable range of 35 and is passed only one time. The spot that gives the maximum number of passes might have a usable range of only 25 but is passed three times. The subject might select a position that gives a good blend of both values (e.g., usable range 30, number of passes 2).

It was not clear from the described experiments exactly how subjects formulated the compromise (i.e., weights, etc.) or if all subjects blended the strategies in the same way. This is a subject for future research.

8.2 *Temporal Placement Strategies*

The primary goal of a tower defense puzzle is to prevent the creeps from reaching the map exit. As described in 8.1 Spatial Placement Strategies, the key is to maximize the combined active time of the set of offensive towers. Spatial strategies maximize active time (a temporal property) by maximizing usable range (a spatial property). While Usable Range is always important, most temporal strategies are based on the idea that time and space are neither proportional nor directionally aligned. In other words, sometimes, decreasing space increases amount of time.

Temporal strategies focus directly on aggregate tower active time or a quantity related to time (e.g., contiguous TEMPORAL ATTACK WINDOW SEPARATION, buffer time, attack window decay). We also include in this category functional properties that have clear temporal effects and actions (e.g., attack window density).

8.2.1 To combat creep time in range

Creeps move through a tower's range area at a fixed speed. Often, they move through the range too quickly for the tower to completely stop all of them.

Strategy TP1: Slow in Range

The most obvious way to solve this problem is to slow the creeps down when they're in a tower's range using the slowing towers. Almost all subjects used this strategy.

Strategy TP2: Swamp

When given a choice of towers, almost every subject spent the majority of their money on offense towers. The only exceptions were players using the swamp strategy. In the swamp strategy, subjects spend all (or almost all) of their money on freezing towers.

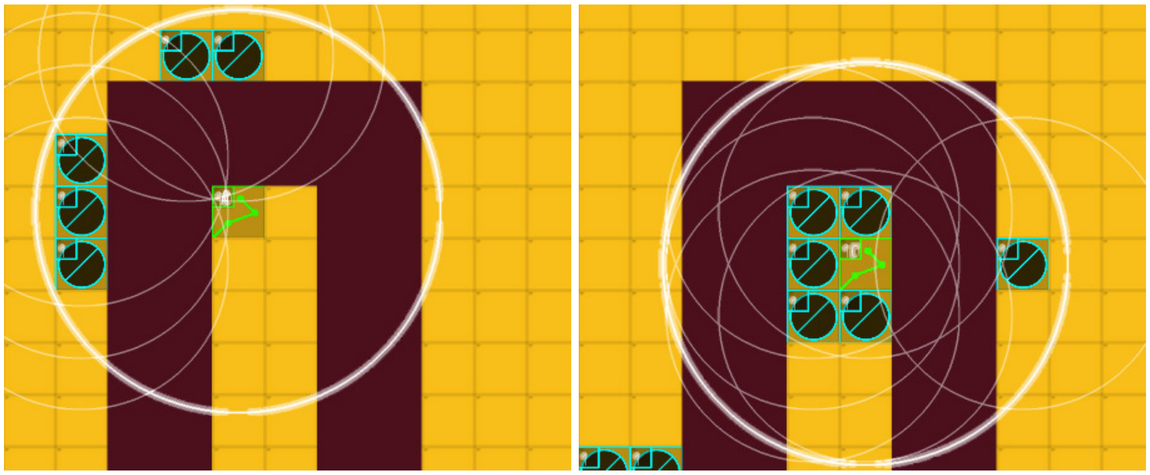


Figure 8.5. Strategy Slow In Range. LEFT: SLOW IN RANGE (3E0100). Blue slowing towers cover part of the range of the green offensive tower. RIGHT: SLOW IN RANGE (3n1000). Blue slowing towers cover all of the range of the green offensive tower.

While slowing towers (including the blue slowing and brown freezing towers) do not do much damage, they do inflict a small amount. A freezing tower can cause one creep to stop moving for two seconds. Because the freezing tower only fires once every two seconds (and due to a slight delay in its firing), a single freezing tower cannot completely stop a creep. Instead, the creep is able to move a small distance every two seconds. Given the extreme difference between the amount of health the creep has and the amount of damage inflicted by the freezing tower, the creep eventually escapes without taking a significant amount of damage.

When placed properly, however, one subject argued that three freezing towers could hold one creep forever. Freezing towers are relatively cheap ($500, out of a budget of $50,000), allowing the subject to buy 100 if they spent their money on nothing else. To stop all 28 creeps, 84 freezing towers are needed. If a creep can be held in place permanently, it doesn't matter how little damage the freezing tower does, eventually it will destroy the creep.

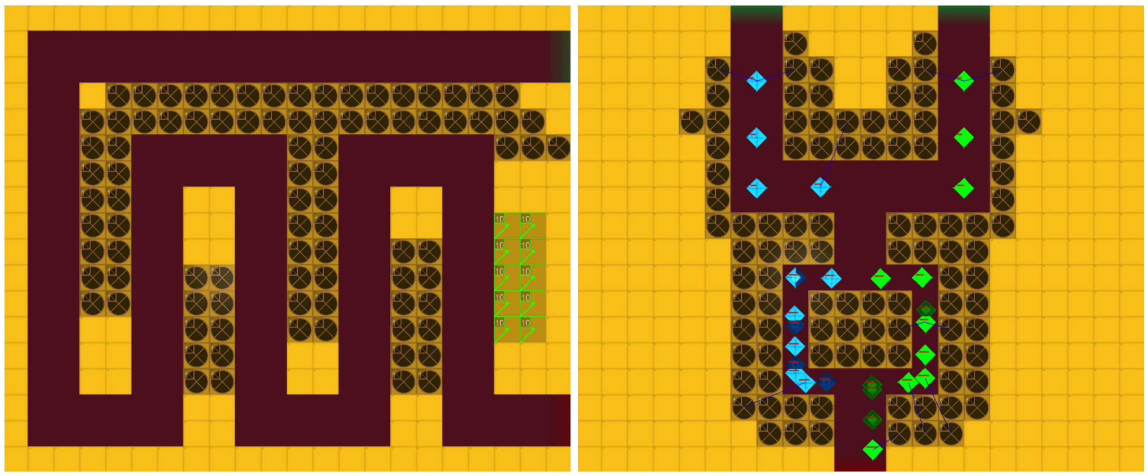


Figure 8.6. Strategy Swamp Attack. Left: Swamp Attack (E0300). 89 level 1 Brown freezing towers are placed along the path. 10 level 10 Green1 fast, weak attack towers are near the exit in a cleanup role. The shortest path is 92 tiles long. The solution earns a score of 28 out of 28. Right: Swamp Attack (E0900). 100 level 1 Brown freezing towers are placed along the path. The longest path is 25 tiles long. The solution earns a score of 16 out of 28.

Swamp attacks do not work on all maps (notably maps with short paths) and do not work well on any map unless properly implemented. This is discussed in more detail below.

To combat synchronized targeting

When a group of creeps enters a freezing tower's range, the freezing tower fires at the first creep and then is unable to fire again for two seconds. During this time, the other creeps in the group move through the freezing tower's range. When two seconds have elapsed, the creeps at the head of the group will have moved through the tower's range. The creeps in the middle of the group will be in range but because of sticky targeting (see glossary) the freezing tower will continue to attack the first creep.

If multiple freezing towers are placed near each other, the first creep to enter one tower's range will likely be the first creep to enter all of their ranges. As a result, all towers near each other will focus on the same creep. This allows the remainder of the creeps to move past all of the towers in the freezing group.

Strategy TP3: Space Out Towers

A single freezing tower cannot hold a creep forever. Multiple freezing towers could be used for each creep but the problem solver does not have the resources to purchase that many freezing towers. For a swamp strategy to be effective, swamp towers need to be formed into groups small enough to hold a single creep permanently (according to subject E0300, this is three towers) but the groups must be spaced apart from each other so that members of the second group do not focus on the same creep as the first group.

Because the groups of towers must be spread out, a necessary precondition for this strategy is that the path is long enough to space all the needed groups along it. The swamp strategy does not work well on maps with short paths (see Figure 8.6).

8.2.2 To combat attack window decay

Assume a tower's range covers two sides of a wall. On each side the tower covers five path tiles for the inside path. When the first creep enters the tower's range, the tower can attack the creep for the amount of time it takes for that creep to move through five spaces. It will have a second chance to attack when the creep moves by the other side of the wall. We call these two opportunities to attack the tower's attack windows.

During the tower's first attack window, the tower can attack the creep for the full time it takes the creep to cross five path tiles. Call this amount of time 5C. While the tower is attacking the first creep, the second creep is moving through the tower's range unharmed. Assume the tower can destroy the first creep in the time it takes the creep to cross two tiles, 2C, and that the second creep is exactly one tile behind the first. After destroying the first creep, the tower can attack the second creep for the amount of time it takes the creep to move four tiles, 4C. When that creep is destroyed on tile three, the third creep will have moved to tile two and the tower can attack that creep for the amount of time it takes to move three tiles, 3C. The amount of time a tower has to attack a creep shrinks the longer the tower attacks a group of creeps. We call this attack window decay. Attack window decay is caused by buffering, the term we use to describe the situation where a creep draws a tower's focus, acting as a buffer that allows other creeps to move through the tower's range unharmed.

The attack window decay lasts for as long as there is a creep in the tower's range. If a tower has a single long attack window, as is the case for U-turns and islands, the attack window can shrink significantly and never recover. This is also a problem if a tower has two attack windows but the distance between them is short enough that creeps enter the second window while other creeps are still in the first. In this case, the creeps on one side of the wall buffer for the creeps on the other.

Attack windows are described in more detail in Section 6.2.2 Attack Windows.

Strategy TP4: Do More Damage

Attack window decay happens because, while a tower is attacking one creep (the buffering creep), other creeps move through the tower's range unharmed. How far the non-targeted creeps can move through the tower's range depends on how long it takes for the tower to destroy the buffering creep. Unless their positions have changed, the next creep in line will be one tile behind. Consequently, if a tower can destroy a creep in the time it takes to cross one tile, the creep will be unable to buffer for other creeps and there will be no attack window decay.

This strategy is closely related to the targeting strategy STRATEGY T1: FOCUSED FIRE.

Strategy TP5: Temporal Attack Window Separation

A spatial attack window is a contiguous region of space where the tower can attack. A temporal attack window is a contiguous block of time that the tower can attack. If the distance between two spatial attack windows is below a certain level, creeps will enter the second area while there are still creeps in the first area, meaning the two spatial attack windows are part of a single temporal attack window. When a creep buffers for other creeps, the size of the per-creep attack window (the time the tower has to attack an individual creep) shrinks with every creep it attacks and the effect lasts for the duration of the temporal attack window.

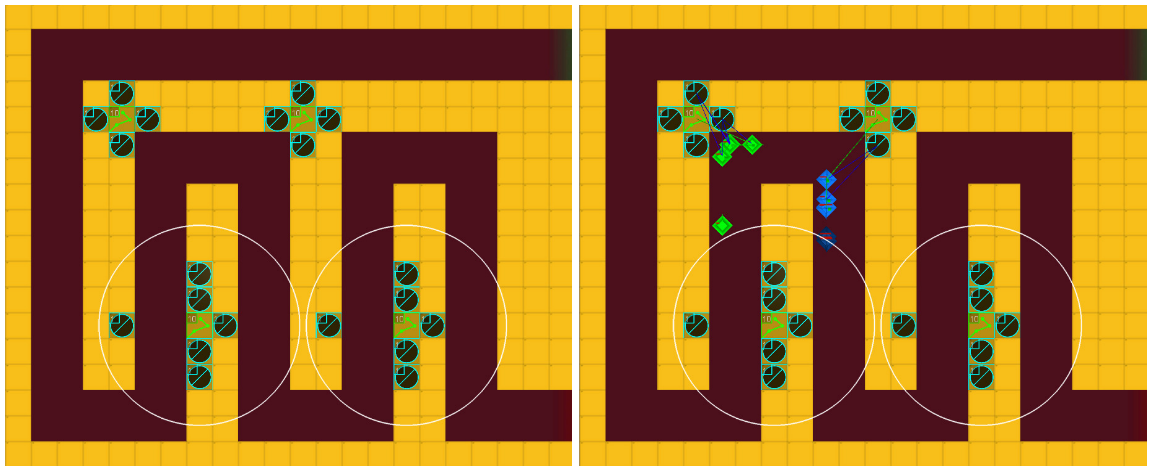


Figure 8.7. Strategy Temporal Attack Window Separation. LEFT: TEMPORAL ATTACK WINDOW SEPARATION (3E0600). Four kill zones, range of bottom two (white circles) shown. RIGHT: Bottom left offensive tower (white circle) has two spatial attack windows. First creep enters the second window just as the last creep exists the first.

In STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION, towers are placed such that their spatial attack windows are sufficiently far enough away from each other (as measured by path length) that all of the creeps will have passed through the first window before any creep enters the next. This creates multiple temporal attack windows. Buffering effects end when there are no more creeps in an attack window, meaning the size of the per-creep attack window returns to normal. Using multiple temporal attack windows therefore decreases the effects of attack window decay, giving the tower more time to attack each individual creep.

This strategy effectively rules out U-Turns and islands. The towers cover less path area but experience less attack window decay. Those who use this strategy are explicitly sacrificing space for time.

The idea here was sufficiently interesting and counter-intuitive that we created test cases to explicitly test the effectiveness of this strategy. Figure 8.8 shows one of the tests. Four level 10 Green3 towers (the same towers used in phase 3) were placed on a double-sided wall. One group was placed at the end where the path formed a U-turn. The other was placed further back so that the group of creeps fully passed through one spatial attack window before entering the second. No slowing towers were used so that the effect of the spatial placement could be measured directly.

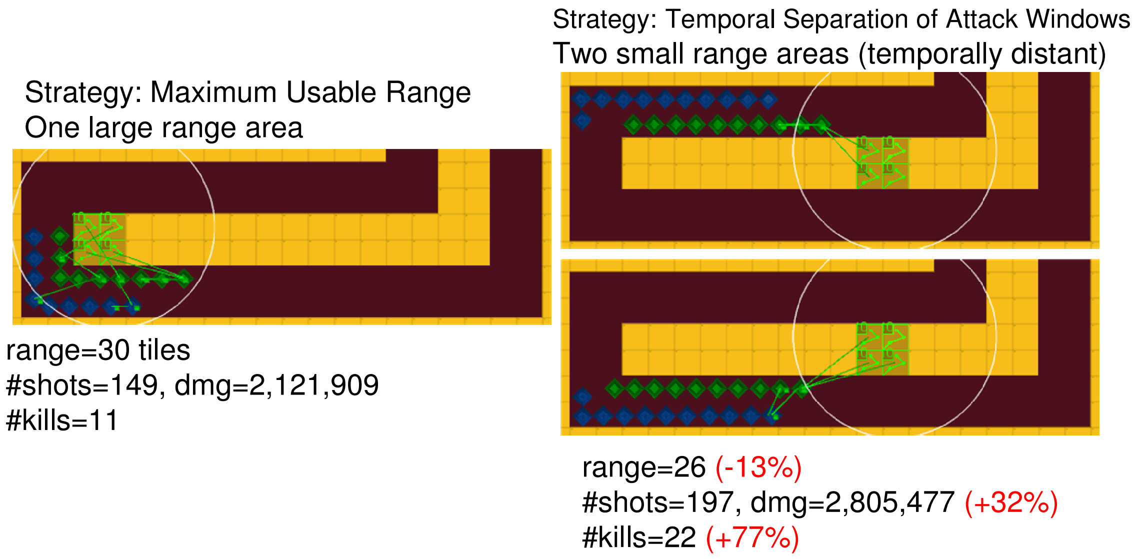

Strategy: Maximum Usable Range
One large range area
range=30 tiles
#shots=149, dmg=2,121,909
#kills=11
Strategy: Temporal Separation of Attack Windows
Two small range areas (temporally distant)
range=26 (-13%)
#shots=197, dmg=2,805,477 (+32%)
#kills=22 (+77%)

Figure 8.8. Impact of Temporal Attack Window Separation. LEFT: Towers placed on a U-Turn cover 30 contiguous path tiles. While the towers focus on the creeps at the head of the group, the others move through the towers' ranges unharmed. RIGHT: Towers placed so that they form two spatially separated sets of 13 path tiles. Towers in this position cover 13% less space but fire 32% more often and score 77% higher.

In the test, the TEMPORAL ATTACK WINDOW SEPARATION strategy significantly outperformed the MAXIMUM USABLE RANGE strategy. The towers covered 13% less range but fired 32% more often. The impact on the score was even more significant – the number of kills increased by 77%, more than twice the increase in the tower active time. In this instance, the significant variance between increased active time and score is due to wounding vs. destruction. Because the towers on the U-Turn had less time per creep due to per-creep attack window decay, the towers were able to wound many creeps but did not have time to fully destroy them. Since the score is based only on creeps that are stopped, not wounded, the score was significantly less than might be expected from the difference in shots fired / active time.

One subject (3E1000) used this strategy but for a different reason. Their focus was on the blue slowing towers. These towers only fire once every two seconds, a very slow rate relative to the amount of time it takes a creep to move through the slowing tower's range. By allowing the creeps to leave for a while before returning, the slowing tower has a chance to "reset", giving it an additional shot. We have already shown that the effectiveness of TEMPORAL ATTACK WINDOW SEPARATION can be explained by the impacts on the offensive tower but the subject might have found a second, complementary reason for why this works. This explanation, proposed late in the experiment, has not yet been tested.

It is worth noting that the required size of the gap between the windows depends on how in synch the two lines of creeps are. As the size of the path synchronization gap grows, the number of positions where towers can be placed under this strategy shrinks. That makes this strategy difficult to combine with strategies such as DIFFERENTIAL SLOWING where the goal is to grow the path gap large enough to completely separate the two lines of creeps.

To combat insufficient room for TEMPORAL ATTACK WINDOW SEPARATION.

The TEMPORAL ATTACK WINDOW SEPARATION strategy places a tower's spatial attack windows such that there is a long enough path between them that the group of creeps has fully left one window before entering the next. If there are 28 creeps and each creep occupies one path tile, this distance must be at least 28 tiles long. It is entirely likely that there is no position where the tower can achieve an inter-window gap of that size. Even when there are, it is possible that the available positions are significantly inferior to positions that have shorter inter-window gaps.

Strategy TP6: Slow Between Attack Windows

As mentioned earlier, two spatially separate attack windows might be a single temporal attack window (i.e., creeps are in both spatial windows at the same time). The goal of the TEMPORAL ATTACK WINDOW SEPARATION strategy is to make sure there is more than one temporal attack window. Specifically, it must take longer for the first creep in a group to move from the start of the first spatial attack window to the start of the second than it takes for the entire group to pass through the first attack window. An obvious way of doing this is to have a sufficiently long path between the two windows. This exploits the correlated nature of time and space – the longer the path between the two windows, the more time it takes to move between them.

An alternate way of achieving the required time between windows is to place slowing towers along the path between the two windows. If all the creeps in the group can be slowed along the entire path, the length of the required path cuts in half.

8.2.3 To combat attack window density

Many subjects noted that, while their tower was attacking one creep, the other creeps in range were moving through the range unharmed. The more creeps that are in range, the more that are protected by the buffering creep. Since the problem is related to the number of creeps in range, the goal of these subjects was to decrease the number of creeps in a tower's range at one time.

Attack window density is complicated by slowing towers. When a puzzle begins, the creeps are arranged equally spaced along a line. This arrangement often changes during game play. Slowing towers can cause a line of creeps to compress (slowing creeps at the head of a line cause them to fall back into the middle; see Figure 8.9) or expand (slowing creeps at the tail of a line causes them to fall far behind). It is important to note that creeps in GopherTD may occupy the same space at the same time.

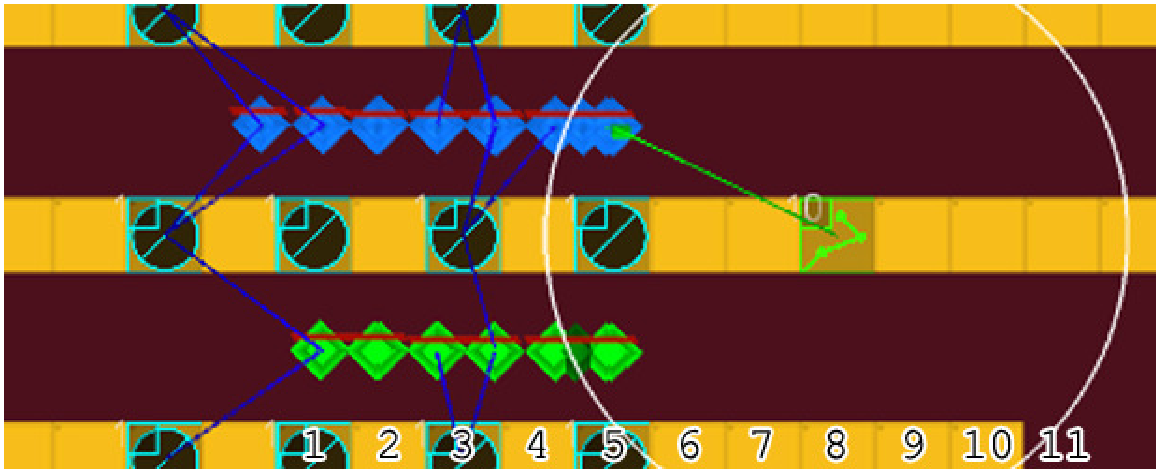


Figure 8.9. Slowing Tower Effect on Attack Window Density. Slowing towers can cause lines to compress, increasing attack window density. A normal line of creeps extends over 11 tiles, giving the green tower shown here 11 seconds to attack. These creeps have been compressed by the slowing towers into five tiles, giving the tower eight seconds to attack, a 27% decrease.

Strategy TP7: Slow Subset to Decrease Density

Each blue slowing tower can slow four creeps. If numerous slowing towers are placed next to each other, the entire group can be slowed. In this strategy, only a few slowing towers are used. The goal is to slow some creeps but not others, causing the group to spread out and thinning out the lines.

Although we didn't extensively test this theory, preliminary tests suggest this strategy is only minimally effective and potentially dangerous. When the creeps at the head of the line are initially slowed, the creeps behind catch up to the first ones and then are slowed, causing creeps to overlap and increasing the density of creeps at those points. The creeps at the end are held back but in tests did not seem to be held back enough to significantly change the line length. The final layout of creeps is chaotic, as shown in Figure 8.10. The danger in this approach is that it appears to be difficult for humans to predict the exact layout of creeps after applying a small amount of slowing. Although the process is deterministic, forecasting the results involves tracking the fire rate of the slowing towers, the duration of the slowing effects and the position of every creep during the active time of the slowing towers.

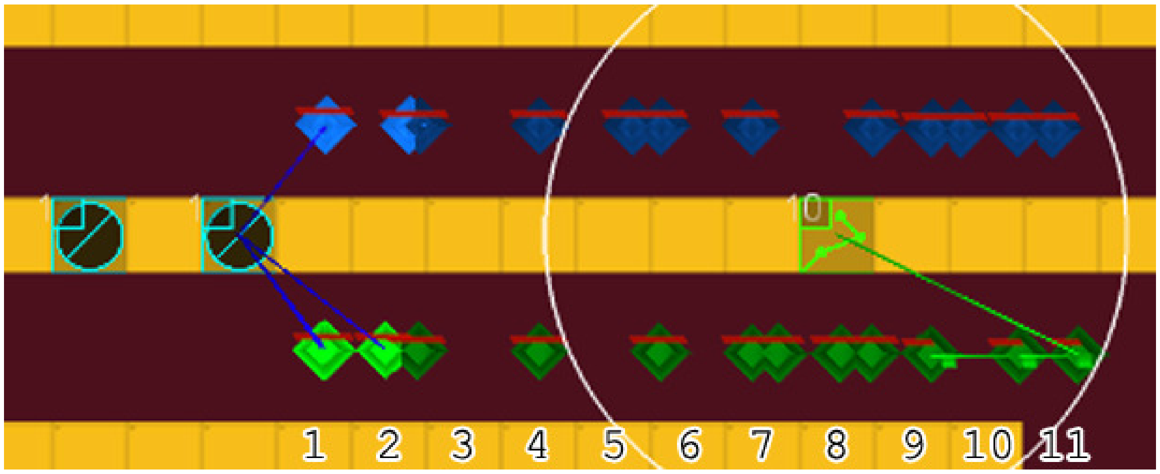


Figure 8.10. Slowing Disturbs Creep Distribution and Density. Slowing a subset of the creeps disrupts the even spacing of creeps, causing some parts of the line to become more sparse, others to become more dense.

Strategy TP8: Exploit Geometry

The EXPLOIT GEOMETRY strategy uses spatial analysis and features combined with reasoning about the flow of time through space to separate the two lines of creeps. This strategy is based on the observation that some paths are longer than others. On some maps, paths have clearly been designed to be longer but even on normal maps one path is either longer to a certain point or overall. The reason for this is corners – the line that is on the outside path around a corner must move further (see Figure 8.11).

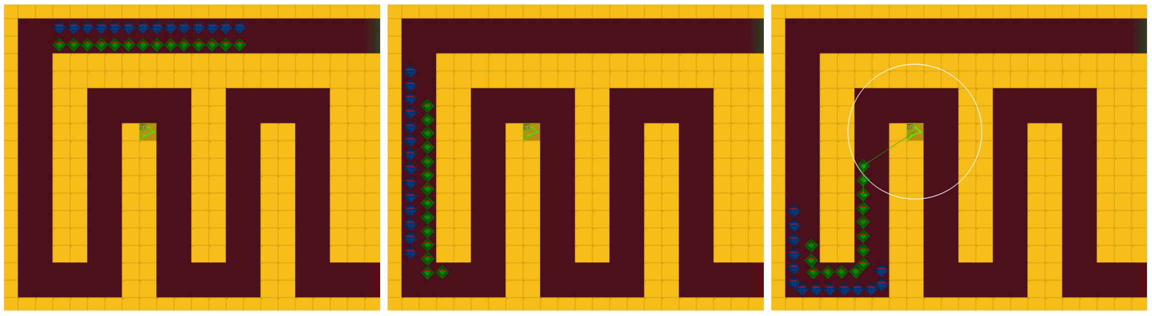


Figure 8.11. Corners Create Path Sychronization Gaps. LEFT: Before moving around a corner, the two lines are in sync. CENTER: After turning a corner, the outer line falls behind 2.5 tiles. RIGHT: After rounding three corners turning the same direction, the inner line is significantly ahead of the outer line.

The key meta-feature used by this strategy is what we are calling a *path synchronization gap* (often referred to as just path gap). This represents how far behind one line is behind the other. When the lines are perfectly aligned (i.e., their heads and tails are immediately next to each other), they are synchronized and there is no path synchronization gap. If one line falls behind by three tiles, the path synchronization gap is three. This strategy places offense towers where the path gap is highest.

As explained elsewhere in this thesis, this strategy works quite well. The level 10 Green3 tower in Figure 8.11 fires 10 times a second doing 1,101 points of damage per shot. It takes an unmodified group of creeps 11 seconds to pass through the tower's range, allowing the tower to fire 110 times and do 121,110 points of damage. Because of the corner in Figure 8.12, left, the outer path is six tiles longer, causing the outer line to fall behind. After the corner, the length of the combined group is 17 tiles. The group takes 15 seconds to move through the tower's range. The tower will fire 150 times, doing 165,150 points of damage, an improvement of 36%.

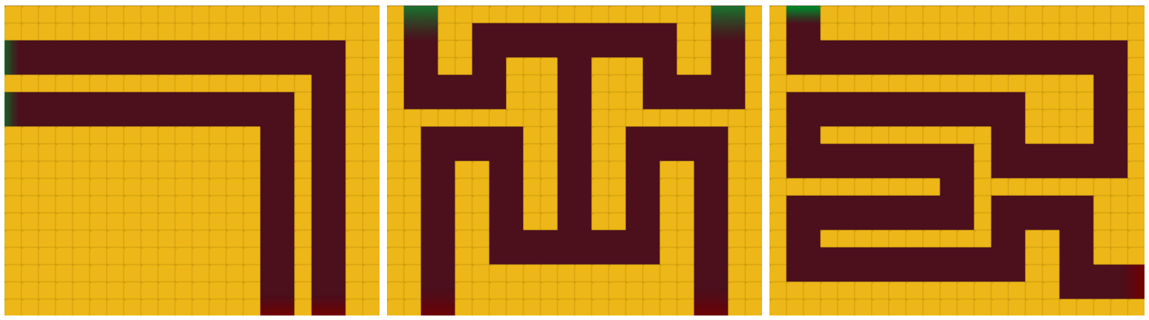


Figure 8.12. Categorizing Maps by Types of Path Synchronization Gaps. LEFT: A Persistent Gap map. CENTER: A No Gap map. RIGHT: A Dynamic Gap map.

From the point of view of the EXPLOIT GEOMETRY strategy, there are three kinds of maps. On Persistent Gap maps, one line pulls ahead and the other line never catches up (Figure 8.12, left). On No Gap maps, the two lines are always in sync (note: this only happens when both creeps follow the same path or mirrored paths; see Figure 8.12, center). On Dynamic Gap maps, the size of the gap keeps changing, with one line possibly in front at some points and the other line in front at others. Dynamic gap maps typically have counter-balancing corners, so that the line that is the outside line on one corner becomes the inside line on another (Figure 8.12, right).

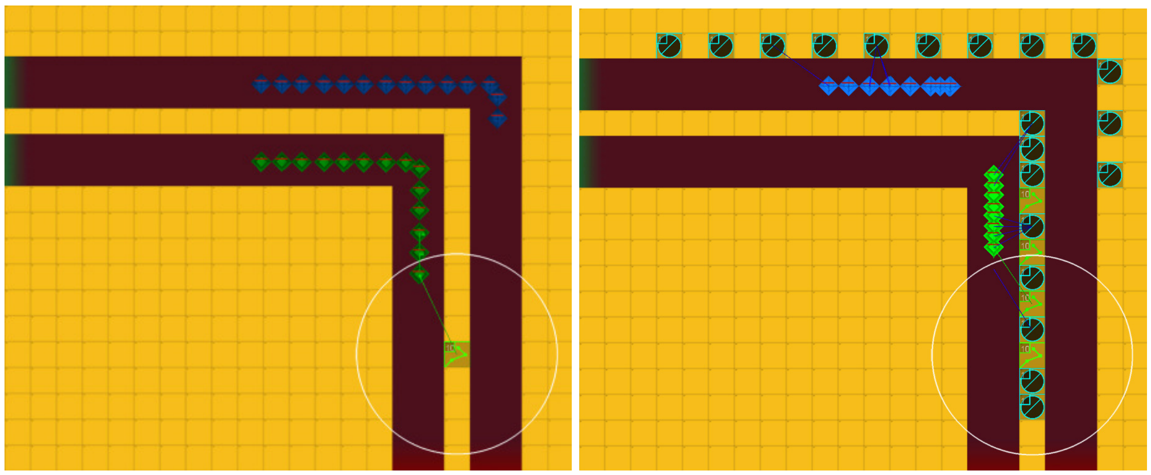


Figure 8.13. Strategies Exploit Geometry and Differential Slowing. LEFT: EXPLOIT GEOMETRY. The upper line of creeps has fallen six tiles behind the lower. RIGHT: DIFFERENTIAL SLOWING (3E0200). The two lines of creeps are completely separated.

It should be noted that very few subjects used EXPLOIT GEOMETRY without also using DIFFERENTIAL SLOWING. Subjects who recognized the path synchronization gap affordance needed for EXPLOIT GEOMETRY almost always immediately recognized that the effect could be enhanced with slowing towers and thus, in an apparent moment of insight, formed the DIFFERENTIAL SLOWING strategy.

8.2.3.a Strategy Family: DIFFERENTIAL SLOWING

The EXPLOIT GEOMETRY strategy uses path synchronization gaps that exist on the map due to the map's geometry. DIFFERENTIAL SLOWING refers to a family of strategies that use slowing towers to create or expand path synchronization gaps.

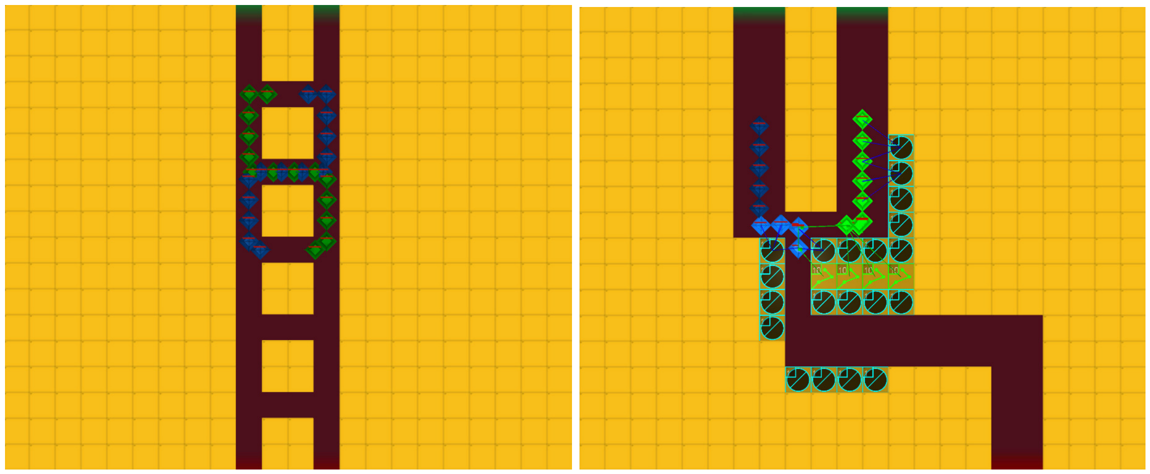


Figure 8.14. Differential Slowing isn't Always Effective. LEFT: On the map Ladder, the creeps weave in and out over each other's paths, making it almost impossible to target a single line. RIGHT: DIFFERENTIAL SLOWING (E0100). DIFFERENTIAL SLOWING seeks to hold one line of creeps back long enough that the other can pull ahead far enough that the two lines are separated. In this solution, because the path between the map entrance and the kill zone is so short, there is not enough time to allow the lines to separate.

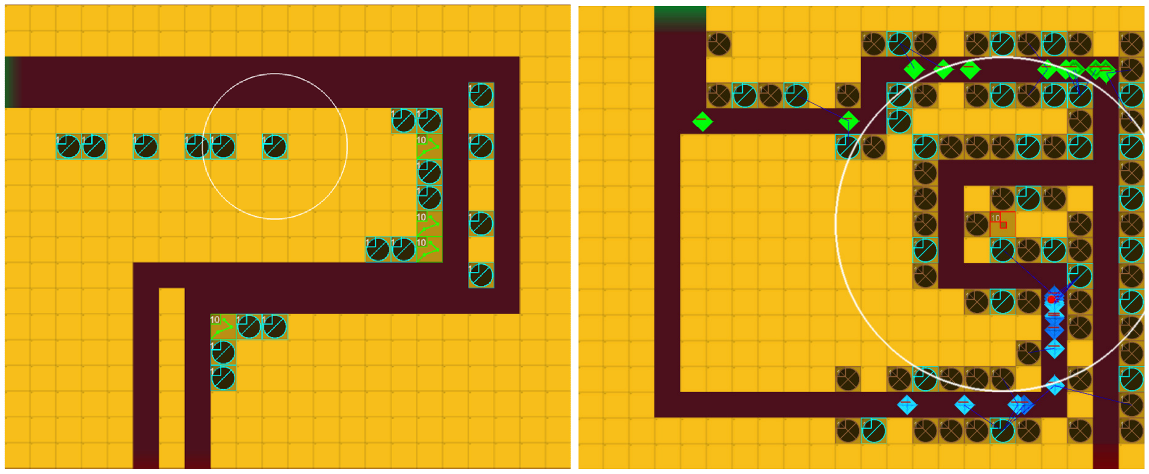


Figure 8.15. Errors Executing the Differential Slowing Strategy. LEFT: DIFFERENTIAL SLOWING without EXPLOIT GEOMETRY (3E0600). The subject slows the line of creeps that is ahead, not behind, reducing rather than increasing the path synchronization gap. RIGHT: (e2500). The subject notices that the two lines arrive at the kill zone at different times but decides this is bad and intentionally places slowing towers to cause them to arrive at the same time.

To first kill zone

A kill zone is an area on the map where offensive towers are setup to destroy or seriously injure creeps. As in EXPLOIT GEOMETRY, the goal is to have one line of creeps enter the kill zone before the other.

Strategy TP9: Permanent Limited View Differential Slowing

The Permanent Limited View strategy causes one line to fall behind the other. It does this by placing slowing towers where the tower is only able to see (and therefore slow) one line. The other line of creeps never enters the tower's range (see Figure 8.16).

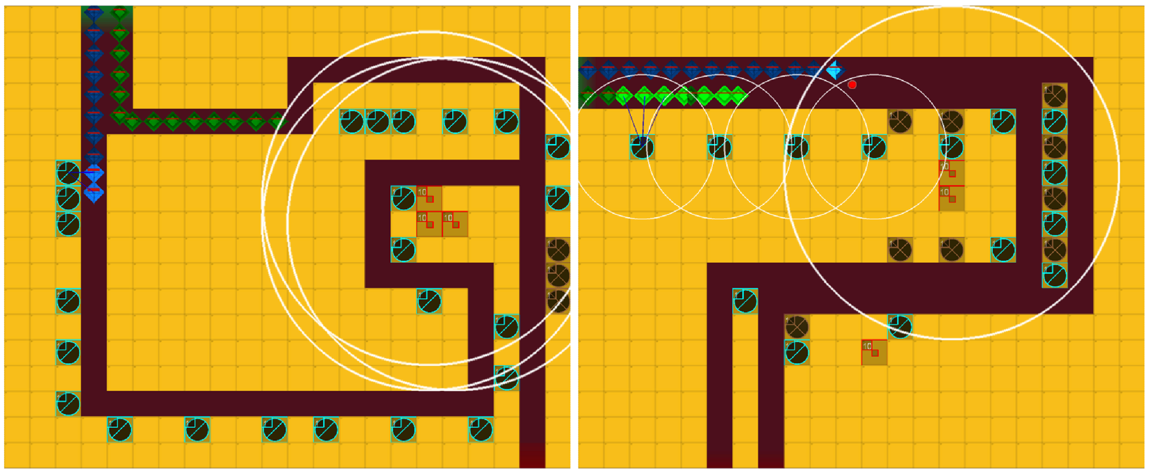


Figure 8.16. Strategies Permanent Limited View Differential Slowing and One Back Positions. LEFT: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING (a0200). The subject slows the left (blue) line, allowing the kill zone offense towers to focus on the right (green) line. RIGHT: ONE BACK POSITIONS (e2600). The subject places the initial slowing towers one tile back from the path so that they can only see (and slow) the bottom (green) line.

Strategy TP10: Pre-Kill Zone Limited View Differential Slowing

Similar to the Permanent Limited View strategy, towers are placed where they can only see one line. Unlike Permanent Limited View, this criterion only needs to hold until the creep enters the first kill zone. If a path passes a slowing tower more than once, the towers are positioned so that they can only see one line on the portion of the path before the kill zone. On the later parts of the path, the slowing towers might cover the path for the same line, the other line or both lines.

A variation of this is to place the slowing towers so that they fulfill two roles, acting as part of the DIFFERENTIAL SLOWING strategy on the first pass and the SLOW IN RANGE strategy on the second.

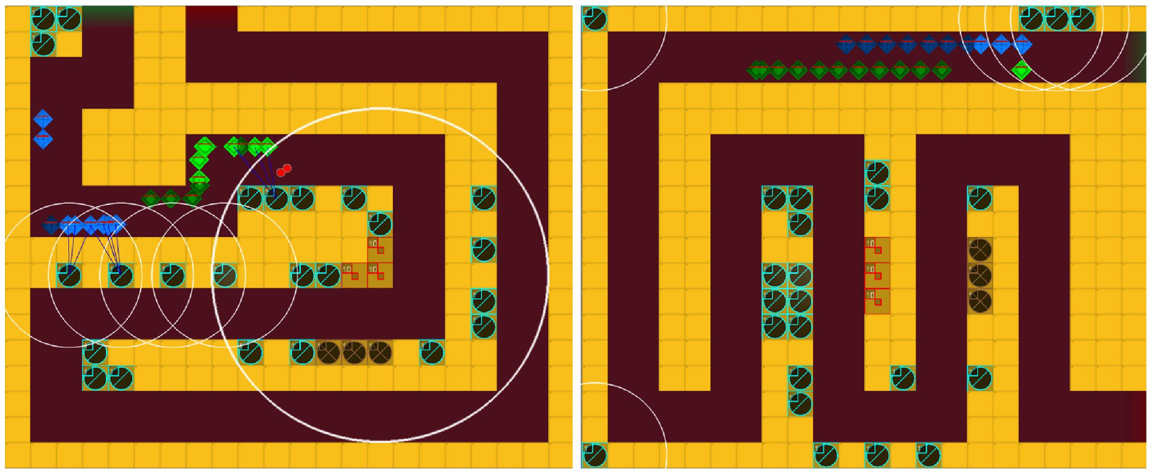


Figure 8.17. Variations of Differential Slowing. LEFT: PRE-KILL ZONE LIMITED VIEW DIFFERENTIAL SLOWING (a0200). The large circle indicates the kill zone. The slowing towers placed before the kill zone are set back one tile so that they can only see (and therefore slow) the left (blue) line before the kill zone. After the creeps move through the kill zone, those same towers can slow both lines. RIGHT: TARGET SELECTION-BASED DIFFERENTIAL SLOWING (a0200). Three slowing towers are placed by the entrance. At this point in time, the top line (blue) will be closer and will slowed, leaving the bottom line (green) to move ahead. An attempt to maintain this path gap is made by placing slowing towers in the corners on the side closest to the slowed line. Unfortunately, this backfires. The bottom line will enter the maintenance slowing towers' range before the top line, causing them to slow the bottom line, reducing the path gap.

Strategy TP11: Target Selection-Based Differential Slowing

The Target Selection-Based DIFFERENTIAL SLOWING strategy exploits the slowing tower's target selection logic to slow a single line.

Two factors control which creeps are slowed, entry time and proximity. The first creep that enters the range will be slowed (assuming the tower is able to fire; slowing towers can only fire once every two seconds so it if has fired recently, multiple creeps might enter the range before the tower selects its target). If multiple creeps enter at the same time, the creep closest to the tower is slowed.

This strategy places slowing towers where they are physically capable of slowing both lines but where the target selection logic will cause them to focus on a single line. These positions are hard to identify because one must know exactly where every creep will be at a given point of time. This is difficult for several reasons. First, corners cause one line to pull ahead of the other. Second, slowing towers slow subsets of creeps, causing the distribution of creeps in each line to be chaotic. Third, offense towers destroy creeps, meaning the creep that might have reached a slowing tower first is destroyed before it gets there.

All of the above factors happen as the creeps move through the map. These complications can be avoided if the creeps are slowed before they move too far through the map. At the map's entry point, the two lines are normally in synch. When true, two factors will cause slowing towers to slow the line nearest them. First, if both lines enter at the same time, the tower selects the creeps closest to them. Second, because the ranges are circular, if both lines are in synch, the line closest to the tower will enter a fraction of a second earlier.

It is important to note that each tower that participates in this strategy causes the furthest/outside line to move ahead of the closer one. The result is that the outside line will enter other slowing tower's ranges first and possibly be closer (because no creep from the inner line has arrived yet), causing later towers to undo the path gap caused by the first ones. According to subject a0200, no more than three slowing towers can be used with this strategy and they must be contiguous (i.e., no gaps between towers). This is normally a one-time option; after the initial slowing, it is difficult to find a second area where the criteria required for Target Selection-Based DIFFERENTIAL SLOWING strategy still hold or can be accurately predicted.

To increase the number of limited view positions

For most subjects, using slowing towers follows several rules: never upgrade (upgraded slowing towers have more range and damage but do not do more slowing), put them next to the path (since their range is small, especially when not upgraded) and, except for those who play DIFFERENTIAL SLOWING strategies, put them so that their range overlaps an offensive tower's range.

On most maps, the two lines of creeps are on adjoining tiles, moving side by side. Even the lowest level towers can see both lines if the tower is placed next to the path. Since these are the positions that most subjects evaluate when placing slowing towers, on most maps, subjects often see no way to execute a limited view-based DIFFERENTIAL SLOWING strategy.

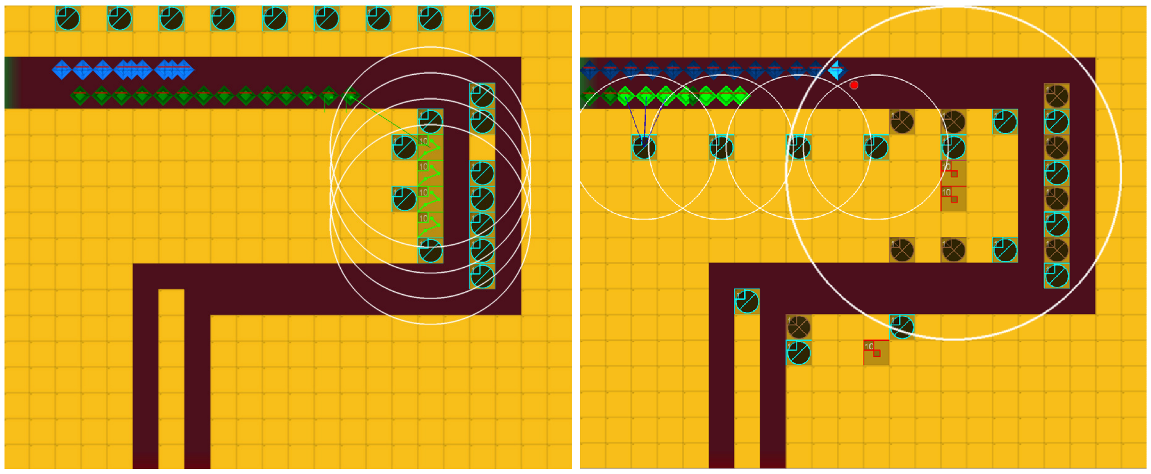


Figure 8.18. One Back Position. LEFT: ONE BACK POSITION (3E0200). The correct way to use a ONE BACK POSITION. Blue slowing towers are placed at the top, one tile away from the path, so that they only slow the top path. To the end of the kill zone, the top path is six tiles longer. Slowing the top path increases the size of this path synchronization gap. RIGHT: ONE BACK POSITION (e2600). The subject places the initial slowing towers one tile below the path, slowing the bottom path. This erases the pre-existing path synchronization gap, increasing the attack window density.

Strategy TP12: One Back Position

A "one back" position is a position that is one or more tiles away from the path (see Figure 8.18). Slowing is normally accomplished using a level 1 Blue slowing tower. Moving this tower one tile away from a path adjoining tile results in the tower's range covering the creeps on the closest line but not the other.

Using ONE BACK POSITIONs sacrifices almost half of the slowing tower's MAXIMUM USABLE RANGE, so this is an instance of sacrificing space for time.

To combat geometry-induced path gaps

Both the DIFFERENTIAL SLOWING and EXPLOIT GEOMETRY strategies seek to create a path synchronization gap between the two lines. When both are present, it makes sense to combine them (i.e., use slowing towers to slow the line that will already fall behind), although not all subjects recognized this (half of the subjects in phase three that used DIFFERENTIAL SLOWING slowed the fast line, causing the path gap to shrink). Sometimes the map geometry causes one line to fall behind but contains no positions where slowing towers can be placed to slow the line that's fallen behind (See Figure 8.19). There are often positions where the line that is ahead can be slowed but this decreases the path gap, running counter to the idea underlying DIFFERENTIAL SLOWING. Under these conditions, it is difficult to play a DIFFERENTIAL SLOWING strategy.

Strategy TP13: Strong Reversal

DIFFERENTIAL SLOWING causes one line to fall back. If the lines are synchronized, a path synchronization gap appears. If the line being slowed is already behind, the gap increases. If the line being slowed is ahead, the gap shrinks and possibly disappears. In the Strong Reversal strategy, the problem solver purposely slows the line that is ahead, applying enough slowing that the gap reverses itself.

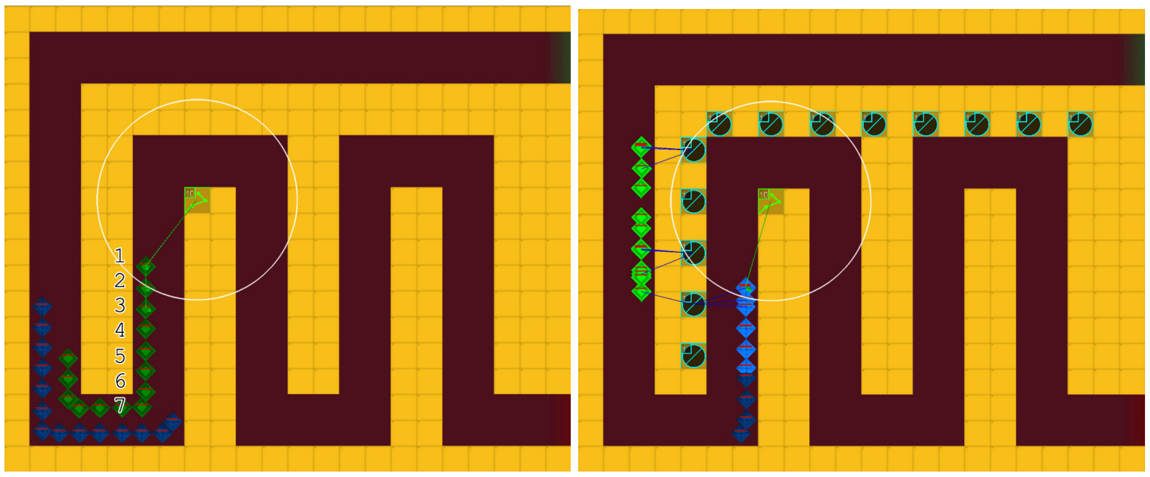


Figure 8.19. Strategy Strong Reversal. LEFT: The bottom (green) line is 6.5 tiles ahead after making three left hand turns. RIGHT: Strong Reversal. Slowing towers are used to reverse the path synchronization gap. The initial +6.5 gap for the bottom (green) line becomes a -12.5, a change of 19 tiles.

To combat Slow In Range undoing Differential Slowing

The SLOW IN RANGE strategy says to place slowing towers so that their ranges overlap with the range of an offensive tower. It is common for one part of a tower's range to cover a kill zone and another part to cover a second area. This second area might be the path leading up to the kill zone. If the problem solver had used DIFFERENTIAL SLOWING to let one line get ahead and a tower used for SLOW IN RANGE slows it down, the positive effects of the DIFFERENTIAL SLOWING strategy are undone.

There are normally several slowing towers in a kill zone and they are often unevenly distributed and appear on both sides of the path. Each slowing tower will slow one to four creeps (depending on how many are in range) and, because they only fire once every two seconds, the closest creeps are not always the first ones. As a result, once the line of creeps has made it past a kill zone, the path synchronization gap, as well as the distribution of the creeps in a given line, will likely change.

Strategy TP14: Avoid Dual-Path Slowing

Some strategies suggest an action to take, others suggest actions to not take. The AVOID DUAL-PATH SLOWING strategy is the latter. In this strategy, problem solvers either avoid placing slowing towers at any position where they will undo the DIFFERENTIAL SLOWING or check the solution after all towers are placed and remove those that interfere with DIFFERENTIAL SLOWING. This can either be done for the area before the kill zone, the area inside the kill zone or both.

Strategy TP15: Reinforce Between Kill Zones

Some subjects placed all their offensive towers in a single kill zone, others distributed their offensive towers across the map, creating multiple kill zones. As discussed earlier, the effects of a DIFFERENTIAL SLOWING strategy are often undone by a kill zone. To maintain or re-establish these effects, a DIFFERENTIAL SLOWING strategy can be implemented between kill zones. If the lines are still fairly separated (which can happen when there is a large path gap going into the first kill zone), maintenance can be done with just one or a few slowing towers. Maintenance can also be done by using slowing towers attached to other kill zones (see Figure 8.7).

Strategy TP16: Kill Zone Jumping

As with the DIFFERENTIAL SLOWING strategies, subject 3E1000's strategy involved having towers focus on one line of creeps at a time. Unlike DIFFERENTIAL SLOWING, they did not use a separate slowing region. The subject grouped their slowing and offense towers in traditional kill zones. The subject noted that if all the slowing towers were on one side, they would slow only the closest line, the other line moving past the kill zone (relatively) unharmed. The subject decided to take advantage of this by using their four offense towers to form four kill zones and spreading those kill zones across the map. When the two lines of creeps entered the first kill zone, one line would be slowed and attacked while the second would move to, and be slowed at, the second kill zone. With the two lines at two different kill zones, the offense towers were able to focus on a single line at a time. When asked to explain their motivation, the subject said they were delaying one line of creeps at an earlier kill zone to help the second kill zone by "remove[ing] temptation".

It was important that the kill zones be as far from each other as possible and that the lines not be slowed between kill zones. This is so that the second kill zone can receive, attack and complete its attack on the first line before the second arrives.

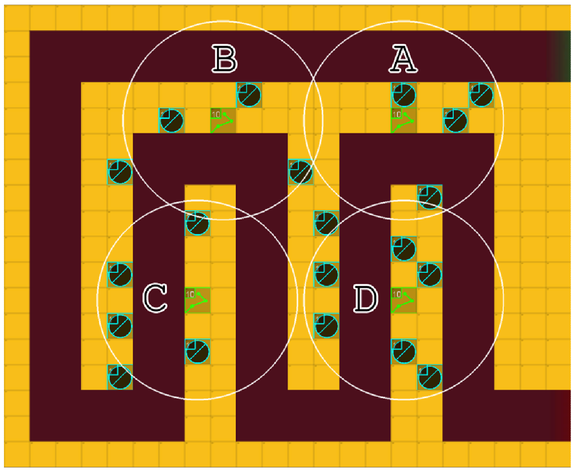


Figure 8.20. Strategy Kill Zone Jumping. KILL ZONE JUMPING (3E1000). Once creeps leave one kill zone, they enter a different one before re-entering the earlier kill zone. Order of kill zones: A, B, C, B, C, D, A, D.

As in the TEMPORAL ATTACK WINDOW SEPARATION strategy, towers were positioned so that they had multiple attack windows. To further combat attack window decay (or, as the subject explained, to give the blue slowing towers "time to reset"), the subject wanted lines that left a tower's first attack window to be attacked by a different tower before entering the tower's second attack window. An example is shown in Figure 8.20. The subject wanted the first attack windows to destroy a subset of the creeps and weaken the rest. This was in preparation for the later attack windows, where the towers could deal with fewer creeps that could be destroyed faster, lessening the impact of attack window decay.

8.3 Satisficing

8.3.1 To conserve mental energy

It is already well understood that humans conserve cognitive energy where possible by using approximations, simplifications, heuristics and sets of specialized, limited-use rules rather than a single, all purpose approach (Smith, John P., Disessa and Roschelle 1993). In this respect, Tower Defense proved no different than other domains that have been studied.

We identified 15 satisficing strategies. Seven are tied directly to spatial features. Specifically, subjects simplified the process of searching for positions that gives a specific tower the MAXIMUM USABLE RANGE to looking for features that are easily recognized by the visual system (e.g., corners, U-Turns, color ratios). One (SATISFICING STRATEGY: TWO PASSES) relates to stopping without examining all options and is also tied to spatial analysis (the number of passes around a position is easily recognized by the visual system). The remaining seven are described here.

Strategy S1: Estimate Value

Many subjects were concerned about their ability to measure the amount of path area in a tower's range. Several subjects moved a tower from place to place or placed two towers at different spots, trying to determine which had the most usable range. Most subjects attempted to determine the value before moving the tower or quickly moved the tower over preferred positions. No subject mentioned or was observed counting the number of tiles within the tower's range, even though it was possible to do so (path area can be estimated using the lines on the adjoining wall tiles).

Strategy S2: Search Subset of Options

Stopping when one has found a good enough option rather than evaluating all options is essentially the definition of satisficing (Simon 1956). Even when subjects directly measure usable range (as opposed to using substitution heuristics such as looking for corners), they did not necessarily look at every position on the map. Some positions were reasonable to exclude, such as those that were far from a path, but some good options were also excluded.

We observed two reasons for stopping a search. The first is that the subject found a position that they believed was the best on the map. This is not quite the same as stopping when they found a position that was good enough – to subjects, "good enough" normally meant finding the best position. The possible quality of a solution, however, was bounded – given that the amount of area covered by a tower's range is fixed and a certain amount must be lost to the wall that the tower is placed on, subjects can tell how close they are to using the maximum amount of usable range. Once a position was found that was close to the highest value, they stopped looking at other positions. If they had continued searching and found a superior position, the position would be better by, at best, a small, fixed margin.

A second reason for not evaluating all options is that many subjects immediately ruled out certain areas of the map, focusing their search on areas they felt the best positions would be most likely to appear in. Excluded areas were often on the borders of the map while core areas were often in the center of the map. It is possible that this is an example of the *Elimination By Aspects* heuristic (A. Tversky 1972). Positions near the exit were especially likely to be ignored, although it is not known whether this is because those positions were on the edge of the maps or for another reason (e.g., subjects had a risk aversion to placing the core of their solution by the exit or subjects searched positions serially along the path and found an acceptable position before then).

Regardless of reason, when asked why one position was chosen, the subjects said it was because it best satisfied their search criteria (normally usable range). When a better alternative was pointed out, the subject typically said they "didn't notice it" and would have selected it if they had.

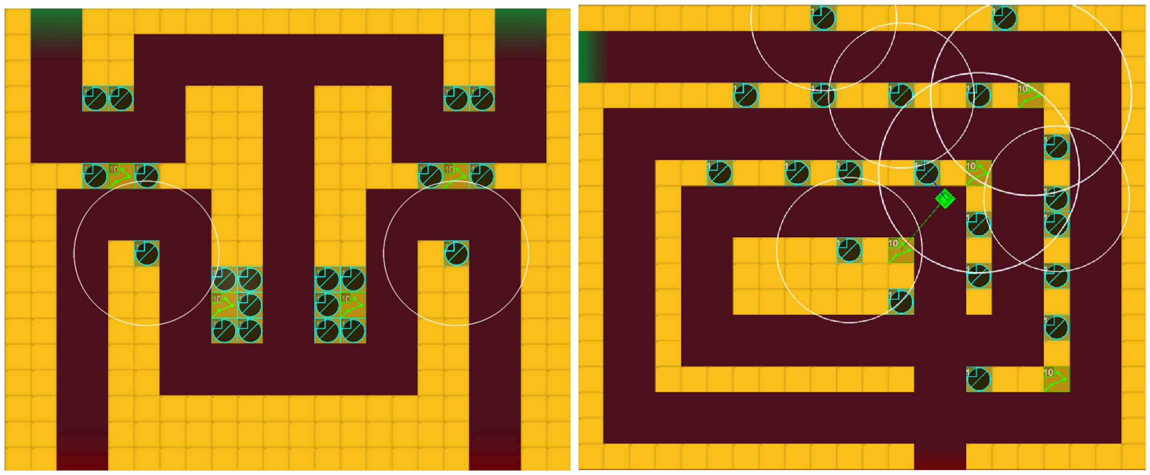


Figure 8.21. Satisficing Strategies. LEFT: SEARCH SUBSET OF OPTIONS (3N0700). Subject prefers to place offensive towers on U-Turns that are two tiles wide. Places two offensive towers on horizontal skinny walls because there are no other acceptable U-turns. Subject doesn't notice the two U-Turns on the sides of the map (white circles), says would have used if they had noticed them. RIGHT: STOP ON ACCEPTABLE SOLUTION (3E0500). "This map is so easy that I don't need to try hard." Quickly places towers in spots that seem reasonable but makes no attempt to evaluate all positions or optimize tower placements.

Strategy S3: Stop on Acceptable Solution

Two of the 16 maps were quite easy, with almost all subjects achieving perfect scores. On these maps, several subjects (although still a minority) declared that there was no need to spend a lot of time on a solution and quickly placed towers, making minimal comparisons between positions and spending no time on optimizing placements.

Strategy S4: Automaticity

Once subjects found an approach that worked for them, they often applied it to new problems without much thought. This allowed them to construct solutions quickly with little cognitive effort. It also caused some subjects to use strategies where they did not make sense.

There were two situations that multiple subjects commented on in interviews. In the first, subjects used ONE BACK POSITIONS (see STRATEGY TP12: ONE BACK POSITION) on maps where it was not helpful (e.g., maps where creeps did not travel side-by-side) (Figure 8.22). In the other, subjects use a DIFFERENTIAL SLOWING strategy on maps where it was not helpful (i.e., lines of creeps could not be significantly separated).

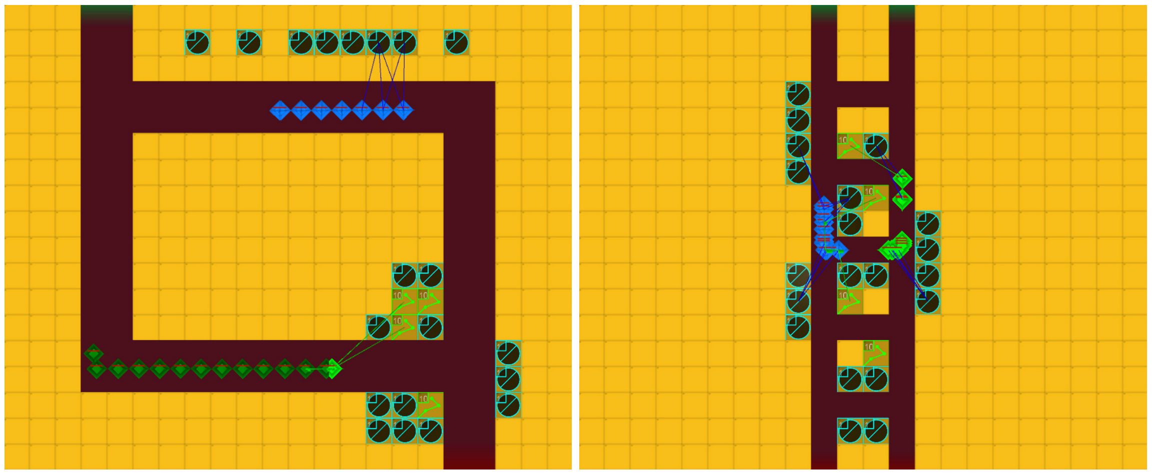


Figure 8.22. Strategy Selection Errors. LEFT: ONE BACK POSITION (3E0700). Subject uses ONE BACK POSITIONS along the top path even though only one line of creeps takes that path. RIGHT: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING (3E0100). The subject attempts, unsuccessfully, to place slowing towers so that they only slow one line of creeps.

ONE BACK POSITIONS sacrifice usable range to target a single line of creeps. If the lines do not travel side by side, the subject sacrifices range for no gain. DIFFERENTIAL SLOWING sacrifices slowing towers that could be used for other strategies (most commonly, SLOW IN RANGE) to separate the two lines. If the lines cannot be separated, the subject sacrifices towers for no gain. There is also an opportunity cost in attempting to use an ineffective strategy when better strategies exist.

Although we did not quantitatively measure this, it appeared that subjects were more likely to catch themselves on errors involving the DIFFERENTIAL SLOWING strategy than the ONE BACK POSITION strategy. This is in one sense counter-intuitive. All other things being equal, ONE BACK POSITION errors should be easier to spot because the criteria for its application are unambiguous. The effects of a particular execution of DIFFERENTIAL SLOWING are harder to predict and the criteria for determining when it is valuable to apply are more subjective. One possible account for this is that the harm in misapplying the ONE BACK POSITION strategy, which is a supplementary strategy in service of the DIFFERENTIAL SLOWING strategies, is relatively minor while the potential impact of misapplying a DIFFERENTIAL SLOWING strategy is more severe, perhaps causing subjects to spend more time evaluating it. Another possibility is that the larger impact of misapplying a DIFFERENTIAL SLOWING strategy caused unexpected results with high valence during training, causing subjects to be more sensitive to that class of errors.

Strategy S5: Remember Solution

The subject attempts to remember how they solved the particular problem the last time they encountered it. In the first two experiments, this was relevant for the minority of test maps that were seen in training. In the 3rd experiment, this was usable for all test maps. Subjects often explicitly asked themselves during testing "now let's see, how did I do this before?" and spent a noticeable amount of time trying to remember the earlier solution and checking each tower placement against their memory.

We consider this unfortunate for two reasons. First, subjects see maps in a fixed order during the training phase and potentially learn new strategies on later maps. By using the solutions they used when they first saw the map, they forgo the option of using any better strategies that they learned later in training.

Second, subjects appeared to have a good memory for "gist" (the basic structure/inter-tower relationships of the set of towers) but not details, resulting in subjects placing towers that matched neither their original placements nor the reasoning behind their original placements. After each map, subjects would often announce that they had misremembered and blame their low scores on their recall problems. Failure to recall exact placement of towers was not the only reason for subject disappointment at their scores. Subjects often had a poor memory for their earlier scores. This leads to two problems. First, subjects claim to experience unfairly high levels of disappointment in their current scores. Second, had subjects remembered their earlier scores more accurately and those scores were low, perhaps they would have been willing to generate new solutions using the strategies they learned later in training.

Strategy S6: Play Concrete Pattern

One subject used a specific arrangement of towers on every map where it was possible. When encountering a new map, the subject did not need to determine which towers to pick or where to place them. The cognitive effort on each map was reduced to simply deciding where the single structure would be placed.

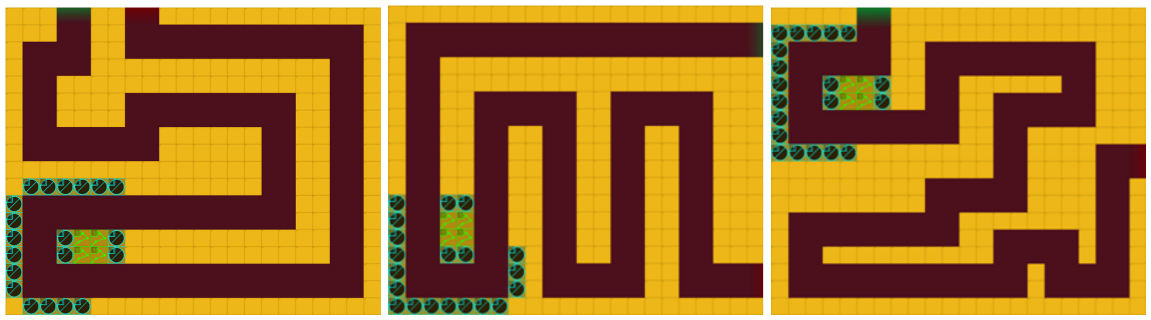


Figure 8.23. Strategy Play Concrete Pattern. Three solution attempts by subject E0100. All three look for a U-Turn around a 4x2 wall segment.

The specific pattern used by subject E0100 required a 4x2 wall section (size-specific, rotation-invariant) at the end of a wall segment to hold four level 10 Green3 offensive towers (middle of the section) and four level 1 Blue slowing towers (ends of the section). This was surrounded in a horse-shoe shaped chain of 16 level 1 Blue slowing towers (see Figure 8.23). The exact layout of this outer portion varied slightly from map to map (for an explanation of why, see STRATEGY O1: INCREMENTAL CHANGE; part of this led to the adoption of STRATEGY E1: AVOID SINGLE SIDED INTERIOR CORNERS).

Strategy S7: Adapt Concrete Pattern

As mentioned in the previous section, subject E0100 used a specific arrangement of towers on every map where it was possible. Where it was not possible, the subject attempted to modify their arrangement to work. Examples are shown in Figure 8.24.

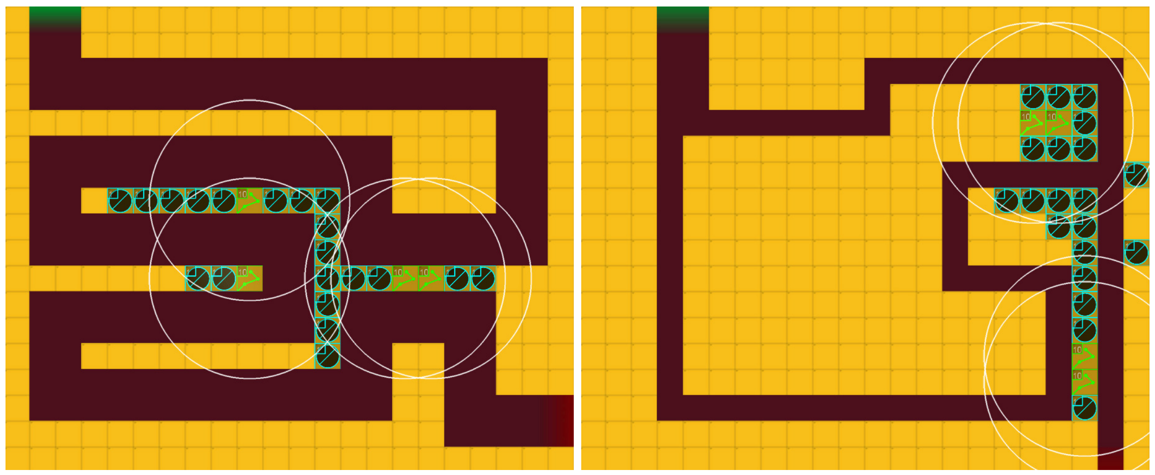


Figure 8.24. Adapting a Concrete Solution. LEFT: ADAPT CONCRETE SOLUTION (E0100). Their normal 4x2 wall segment feature doesn't exist so the pattern was adapted to the existing geometry. The range of the four offensive towers is shown by the white circles. RIGHT: ADAPT CONCRETE SOLUTION (E0100). Their normal 4x2 wall segment feature doesn't exist so the pattern was adapted to the existing geometry. The range of the four offensive towers is shown by the white circles.

Adaptation of the pattern was neither quick nor certain – the subject moved towers from position to position, trying to find the spot that best matched the intention behind the original pattern. This required the subject to recall why they had created the original pattern they were modifying.

In other tower defense games, the subject preferred to use area of effect towers, which hit all targets in range (see Area of Effect in the glossary for more information). The subject tried to replicate this ability by using several Green3 towers (which attack three targets simultaneously) and placing them as close together as possible to form one big meta-tower. This resulted in placing them in a 2x2 formation in the center of the 4x2 wall segment. To make the pseudo-area of effect tower more effective, its entire range was covered by Blue slowing towers.

The subject had four goals – group the offensive power into as small an area as possible, place it where it had the most usable range, slow the creeps while they are in the tower's range and place everything close together so that the same set of slowing towers could do all the slowing work for all of the offensive towers. While the subject could state their goals, it was not clear if they had always held them in mind (i.e., there was a dual representation of the strategy, both as a concrete pattern and as a set of goals) or re-created the goals by studying the concrete pattern.

Where the subject's concrete pattern could not be used, the subject moved back to the goals and created a novel solution. The same set of towers was always used. Without the 4x2 block, the four offensive towers could not be grouped into a tight 2x2 cluster. In all instances, the subject chose to break the four towers into pairs, creating mini-area of effect towers (see Figure 8.24). The pairs were placed close enough to each other that they could share slowing towers but far enough away that all creeps in both lines could be slowed between the offensive towers.

Because only one subject used and adapted a concrete pattern, it is difficult to draw any general conclusions about the nature of solution adaptation in a spatiotemporal domain such as this. Despite this, several items appear to be interesting. The subject chose to modify their pattern rather than create a new solution. The adapted solution looked nothing like the original pattern.
The subject used the same pieces (towers) in the same roles. The goals behind the original pattern were available, either by recall or reconstitution, and explicitly addressed. Adaptation was based on the goals behind the original solution rather than the structure. We believe this topic is worth further investigation.

8.4 Targeting Strategies

Targeting strategies involve placing towers in a way that controls which creep they target. This is typically done to achieve a goal achievable by but unrelated to targeting.

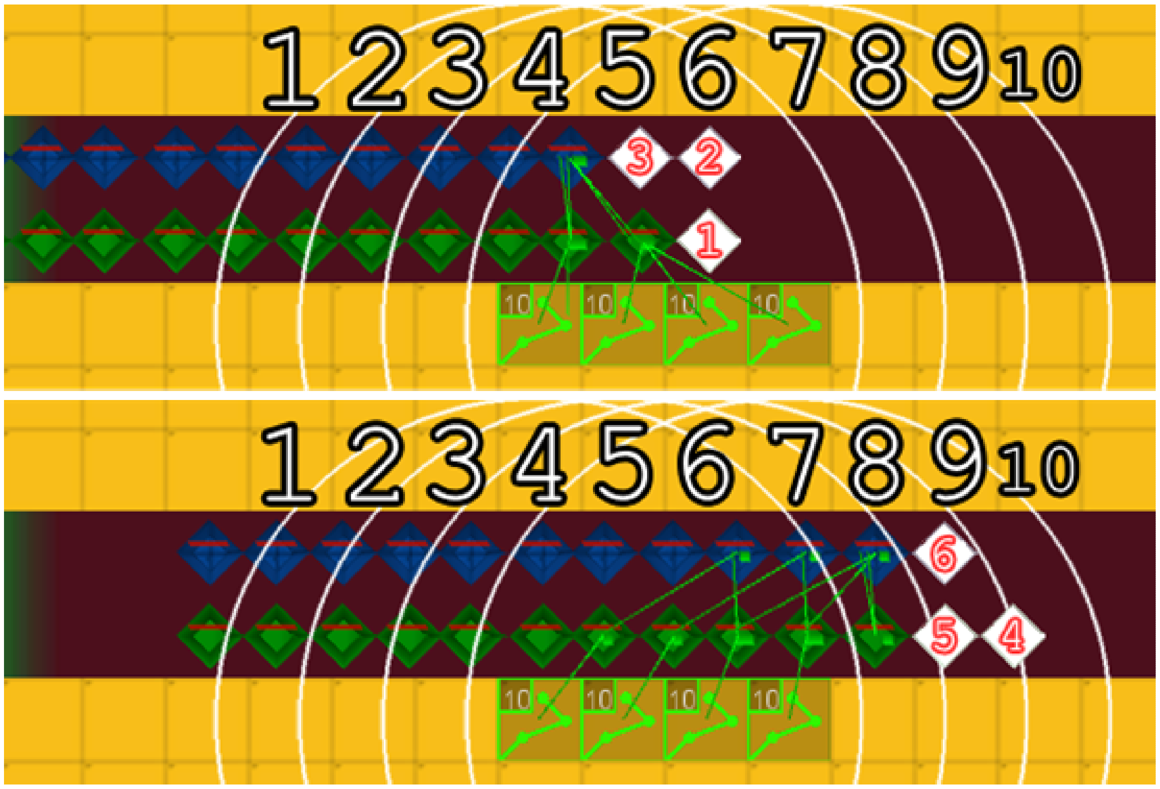


Figure 8.25. Creeps Move through the Towers' Range while being Attacked. White circles indicate each tower's range. White diamonds shows where creeps are destroyed. Top: First creeps are destroyed at position 6. The following creeps have moved to position 5. Bottom: The second set of creeps is destroyed at position 10. The following creeps have moved to position 8.

8.4.1 To Combat Temporal Attack Window Decay

While the exact number varies, each tower can attack a limited number of creeps at one time (most of the following examples show Green3 towers, which can attack three creeps at a time). Towers attack the creeps closest to them, starting as soon as the creeps enter the tower's range. While those creeps are being attacked, the other creeps are able to move through the tower's range without being attacked. We say that the creeps being attacked are *buffering* for the other creeps, preventing the other creeps from being harmed.

A tower can only attack a creep while it is in range. In Figure 8.25, the combined range of the four towers is 10 tiles wide. The amount of time the towers have to attack the first creep is equal to the amount of time it takes that creep to move through those 10 tiles. We refer to this time period as the towers' *per-creep temporal attack window*. In the example, the amount of time it takes to destroy the first creeps is equal to the amount of time it takes the creep to move six tiles.

When the towers have destroyed the first creeps, the next closest creeps have moved to position five. While the first creeps had to make it through 10 tiles to survive, the second set only have to move six (tiles five through 10, inclusive). The towers' per-creep temporal attack window drops from 10 to six. We refer to this as *attack window decay*. When the second set of creeps is destroyed, the next set of creeps has made it to position eight. The towers' per-creep temporal attack window drops from six to three.

As the per-creep temporal attack window shrinks, the total amount of time the tower has to attack shrinks. We refer to this total time as the tower's *active time* and it is directly related to the number of shots the tower makes and the amount of damage the tower does. The following strategies attempt to combat buffering (and as a consequence, attack window decay) and to increase a tower's active time.

Strategy T1: Focused Fire

Towers in GopherTD attack the closest creep and then stay focused on that creep until it leaves range. The goal of FOCUSED FIRE is to group the towers as closely as possible so that the creep that is closest to one tower is closest to all of them. An example is shown in

Figure 8.26. If all towers focus on the same creep, that creep will be destroyed sooner, thus limiting the creep's effectiveness as a buffer for the other creeps. If two towers attacked different creeps, both towers would do just as much damage as if they had worked on the same target (with the exception of overage, discussed later) and possibly destroy both creeps, but they will take longer to do so, allowing the following creeps to move further through the tower's range and thus decreasing the tower's active time and effectiveness.

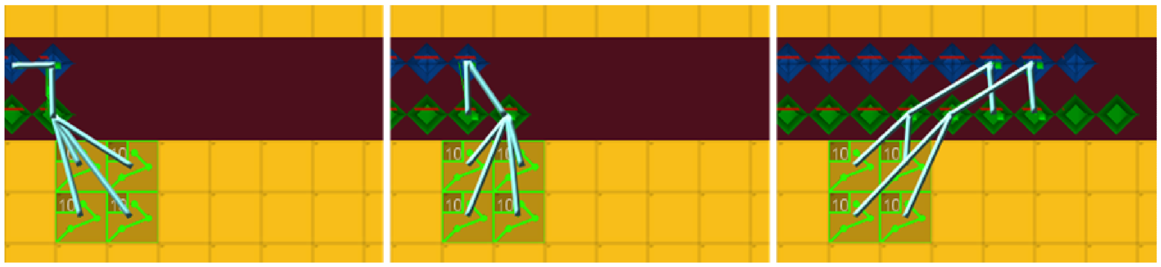


Figure 8.26. Targets Chosen by Four Towers Grouped for Focused Fire. Green3 towers hit three targets in a chain, first hitting the closest target then bouncing from there to the creep closest to the first and then bouncing to the creep closest to the second. Shots are highlighted in white. Because the towers are so close together, they tend to select the same targets.

If more than two towers are involved, the tower grouping will look like a ball. Some of the towers might be adjacent to a path but the others are likely to be removed from the path, thus decreasing their usable range. This strategy therefore potentially sacrifices space for time, trading usable range for a possible increase in a tower's activation time.

There are two alternatives that might lead to synchronized targeting without sacrificing usable range. A *dense chain* places all the towers side by side (Figure 8.27, bottom). The first creep to enter the first tower's range will be the first creep to enter the next tower's range, causing each tower to attack the same creep. This is only guaranteed on the first creep. If the creeps manage to reach the second tower, the creep that is closest to each tower will be different.

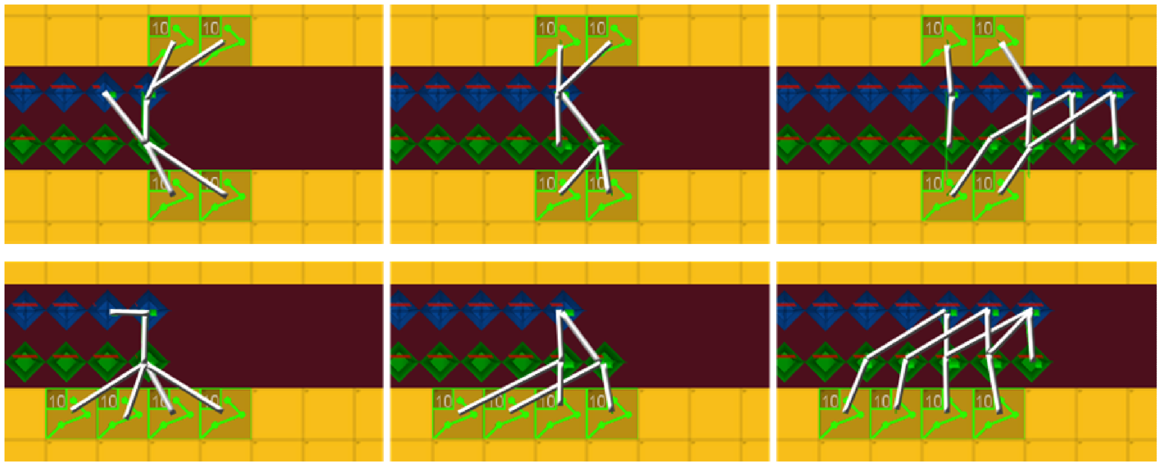


Figure 8.27. Targets Chosen by Different Tower Layouts. TOP: CONCENTRATED OPPOSING FIRE. At first, all towers target the same creeps. As time goes on, different towers choose different targets. BOTTOM: DENSE CHAIN. As with the top row, towers initially focus on the same creep but separate as time goes on.

CONCENTRATED OPPOSING FIRE places the towers as close as possible while still covering both sides of a path (Figure 8.27, top). If there is only one line of creeps, this will act like FOCUSED FIRE. In most instances, though, there are two lines and the towers on each side of the path will pick different targets. In phase three, subjects were required to use Green3 towers. These towers attack three targets, one of which is often on the other side. For this reason, the towers sometimes attack the same set or subset of creeps.

Phase three of GopherTD attempted to measure the value of several strategies. FOCUSED FIRE was not used often enough to do statistical analysis. Even so, it was used by two of the top scoring subjects in phase three and by several subjects in earlier phases. The strategy was explicitly described by one of the top scoring subjects (a0200). For this reason, we conducted a series of synthetic tests to determine how much the FOCUSED FIRE strategy added to the player's score. Unexpectedly, the answer appears to be nothing.

Figure 8.28 shows four different tower layouts: MAXIMUM USABLE RANGE, DENSE CHAIN, FOCUSED FIRE and CONCENTRATED OPPOSING FIRE. Each layout uses the same type and number of towers. To simplify the analysis, no slowing was used. All towers are in approximately the same area with the exception of MAXIMUM USABLE RANGE. The map, Up 'n Down, is one of the maps used in all three phases of the GopherTD experiments. Results are shown in Table 1.1.

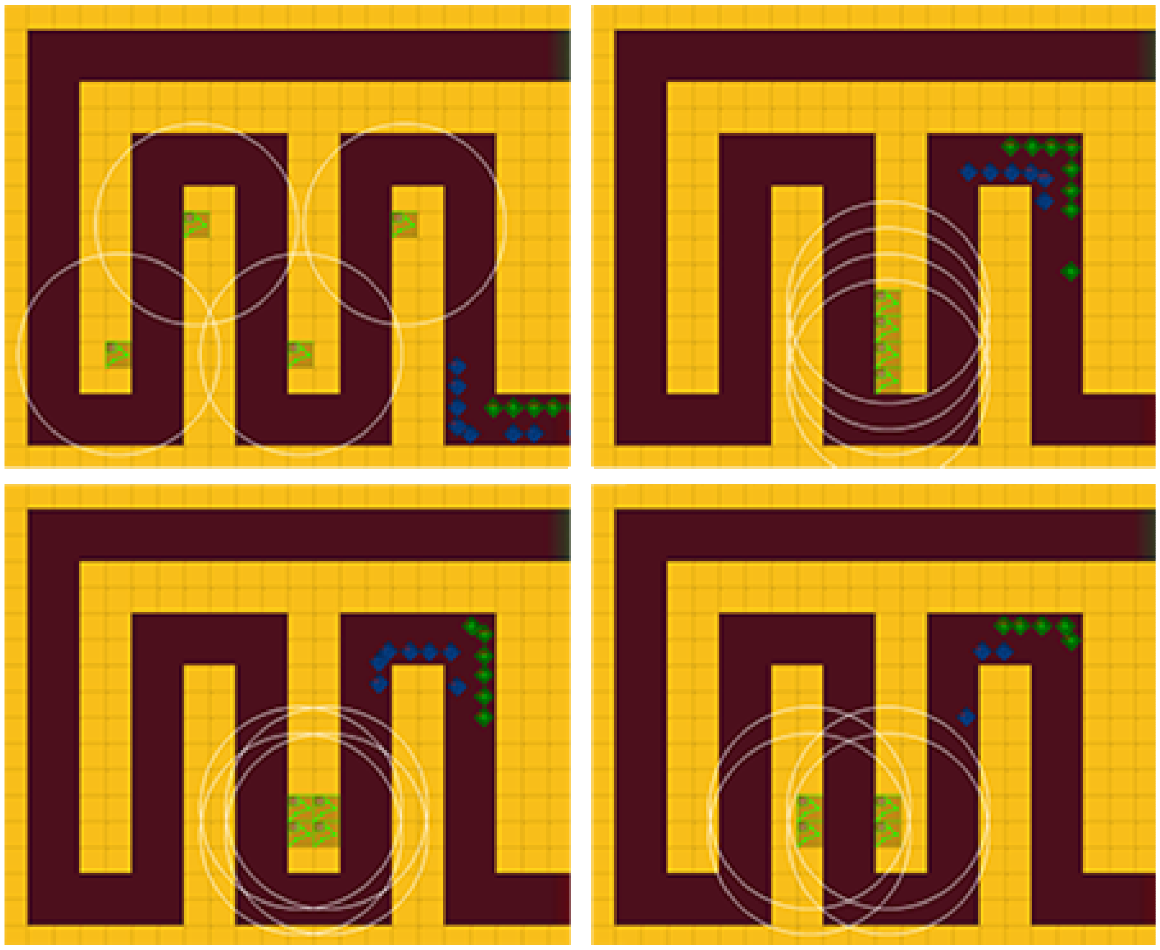


Figure 8.28. Common Methods of Distributing Towers. TOP LEFT: MAXIMUM USABLE RANGE. TOP RIGHT: Dense Chain. BOTTOM LEFT: FOCUSED FIRE. BOTTOM RIGHT: CONCENTRATED OPPOSING FIRE.

	Total Range	Score
Maximum Usable Range	120	12
Dense Chain	111	14
Focused Fire	116	15
Concentrated Opposing Fire	111	20

Table 8.1. Usable Range and Score For Different Tower Layouts. Towers are four level 10 Green3 fast attack towers. Layouts shown in Figure 8.28.

FOCUSED FIRE outperformed MAXIMUM USABLE RANGE but significantly trailed CONCENTRATED OPPOSING FIRE. We cannot currently explain why. One possibility is that there is something inherently wrong with FOCUSED FIRE. Another possibility is that there is something unusual about this map that affected the score. Variables of potential interest include:

- **Line Focus**. As will be shown in Section 8.4.2 To Convert Wounds to Kills, dividing towers across targets results in more creeps being wounded and fewer being destroyed. By this criterion, CONCENTRATED OPPOSING FIRE should do the worst because two towers are closest to one line of creeps and the other towers are closest to the other line. Despite this, it was highest scoring configuration.
- **FOCUSED FIRE**. The goal of grouping towers together is to have all towers be closest to, and thus focused on, the same creep. This is why opposing fire configurations should do poorly and FOCUSED FIRE configurations should do well. However, these configurations do not necessarily result in synchronized targeting. In Figure 8.28, FOCUSED FIRE splits damage across two sets of creeps by the third set of kills while in Figure 8.28, DENSE CHAIN splits into two groups on the second set of kills and four on the third. Despite this, the multiple bounces of the Green3 towers caused the sets to overlap, partially mitigating the effect. A similar effect happens for CONCENTRATED OPPOSING FIRE.
- **Usable Range**. All other things being equal, the configuration that covers the most path space has the highest activation time, does the most damage and gets the highest score. In this instance, however, the configuration with the most usable range (MAXIMUM USABLE RANGE) did the worst and the solution with the

least usable range (CONCENTRATED OPPOSING FIRE) not only scored the highest, it scored 67% higher than the solution with the most usable range.

- **Path Synchronization Gap**. Like most maps used in GopherTD, one line of creeps was ahead of the other. In this case, the specific position used was selected because it is where the path gap is the lowest (initially three, although it grows while rounding the U-Turn). The MAXIMUM USABLE RANGE configuration included two places where the gap was the largest (nine). Two of the towers in the CONCENTRATED OPPOSING FIRE configuration also cover an area where the path gap is largest, although only for part of those tower's ranges.
- **Number of Temporal Attack Windows**. The CONCENTRATED OPPOSING FIRE configuration has two contiguous temporal attack windows, although the second window represents only 25% of the total usable range covered. The other configurations have only one.
- **Tower Sensitivity**. Perhaps the value of these configurations depends on the type of tower used. An example done with a qualitatively different type of tower is discussed in the next section.

To more deeply study the effects of buffering and attack window decay, another set of tests was done on five tower configurations. All towers were placed on a straightaway with no path synchronization gap and no difference in usable range (other than FOCUSED FIRE) or number of temporal attack windows. The creep positions at two points in time are shown in Figure 8.29. Results are reported in Table 8.2.

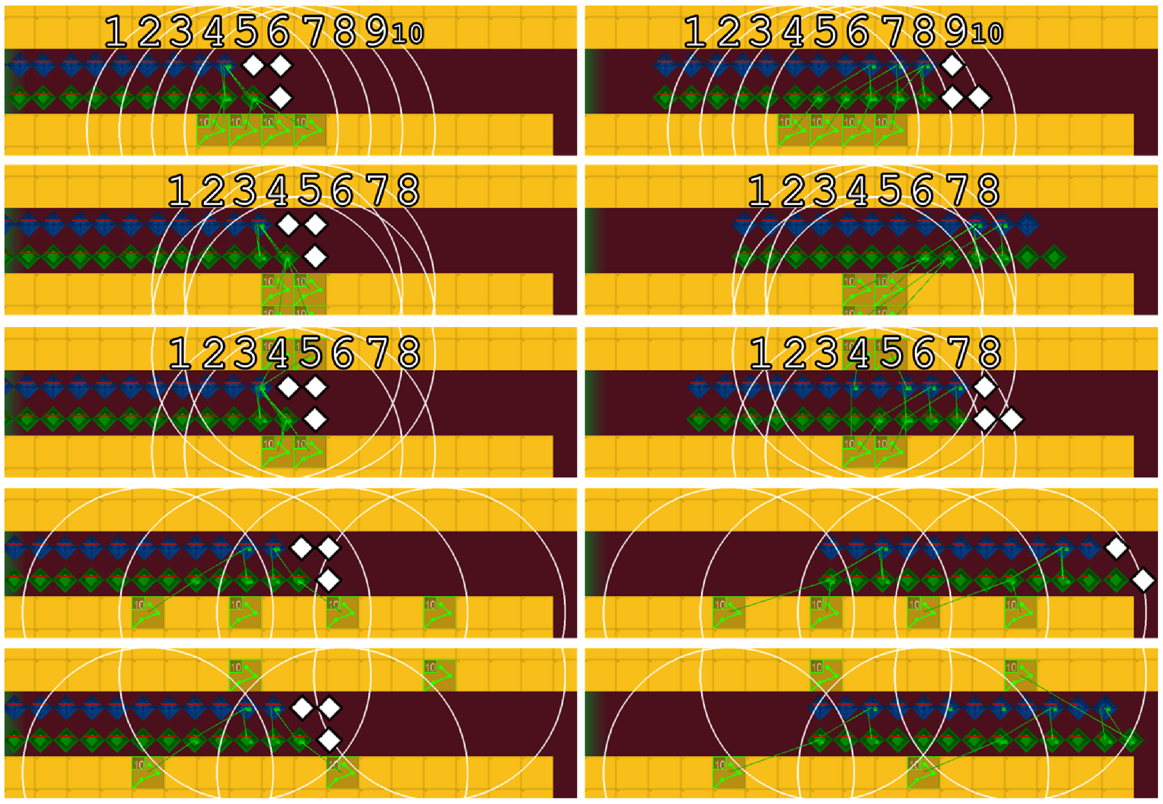


Figure 8.29. Comparison of Different Tower Arrangements. Green3 towers attack three creeps at once. White creeps indicate where a creep died. ROW 1: DENSE CHAIN. The first creeps are stopped at position 6, the second at position 10. ROW 2: FOCUSED FIRE. The first creeps are stopped at position 5, the second escaped severely damaged but alive. ROW 3: CONCENTRATED OPPOSING FIRE. The first creeps are stopped at position 5, the second at position 9. ROW 4: SPREAD OUT. The first creeps are stopped at the third tower, the second at the edge of the fourth tower's range. ROW 5: SPREAD OUT OPPOSING. The first creeps are stopped at the third tower, the remainder escape.

The first row shows four level 10 Green 3 towers organized as a Dense Chain. The combined length of their attack windows is 10 tiles. Because the ranges overlap, the towers can apply four times as much power to the center tile as the first and last ones, which are covered by a single tower.

The amount of time the towers have to attack the first creep is equal to the amount of time it takes for the creep to move from the start of the range to the end, 10 tiles away. The size of the combined per-creep temporal attack window is 10. The first creeps are destroyed when they reach tile six. The next creeps are at position five. The time to attack those creeps is the time it takes for them to move six tiles (five through 10, inclusive). The second set of creeps is stopped at position nine. The following creeps are at position eight. The size of the attack window now drops to three, the time it takes for those creeps to move through the last three tiles. The size of the combined per-creep temporal attack window drops from 10 to six to three.

The second row shows the towers organized as FOCUSED FIRE. The combined length of their attack windows is eight tiles. Although shorter than the range in Dense Chain, it is more powerful – almost every area in range can be reached by all four towers. The initial size of the per-creep temporal attack window is 8. The first creeps are stopped at position five, the next creeps having made it to position four, one tile earlier than in Dense Chain. The new size of the attack window is five. This is enough time to wound the next creeps so severely that they have only a few health points when they exit the range but not enough time to destroy them. When the creeps move out of range, the towers select the towers that are closest to them, which are those in the middle of the line. At this point, the towers select creeps directly above them, resulting in the two columns of towers selecting different creeps. The creeps targeted by the left towers must move five tiles, the creeps targeted by the right towers must travel four. The size of the window remains stable, although it means that many creeps at the head of the list escape unharmed. Note: measurements are approximate, with fractions of tiles rounded on a per-tile basis. Since the range areas are circular, the top line is in range slightly less than the bottom line. These small variances explain why the towers were able to destroy the first creeps in five tiles but not the second.

The other configurations are similar, with longer ranges doing less damage per tile (since fewer towers can reach that position) and vice-versa.

	Size of TAW 1	Size of TAW 2	Size of TAW 3	Usable Range	Score
Dense Chain	10	6	3	52	6
Focused Fire	8	5	4/5	48	4
Concentrated Opposing Fire	8	5	2	52	6
Spread Out	16	8	1	52	5
Spread Out Opposing Fire	16	8	varies	52	3

Table 8.2. Effect of Focused Fire on Temporal Attack Window Size. Using the position of the lead creeps in Figure 8.29, we can determine the size of the per-creep temporal attack window. The size of Temporal Attack Window (TAW) 1 is the amount of time the combined set of towers has to attack the first creep. TAW 2 is the size of the window after the first creep has been destroyed. TAW 3 is the size of the window after the second set of creeps is destroyed or moves out of range.

After removing the effects of usable range (with the exception of FOCUSED FIRE), path synchronization gap and number of contiguous temporal attack windows, it is still not clear why the results are what they are. Opposing Fire is a facet of both the highest and lowest scoring configurations. It is hypothesized that the lack of targeting synchronization is why the Spread Out configurations slightly under-perform. It is also hypothesized that the poor showing of FOCUSED FIRE, which was right on the verge of destroying three other creeps, is due to the slight deficit in usable range.

8.4.2 To Convert Wounds to Kills

The more time a tower is active, the more damage it can do. The more damage it does, the more creeps it can destroy. But doing more damage is not guaranteed to stop more creeps. The number of creeps destroyed depends on how much total damage is done, how the damage is distributed and how much is wasted (discussed in the next section).

Given the choice between many wounded creeps or a few destroyed ones, the latter is preferable as wounding creeps does not increase the player's score. If damage is not distributed properly, one might end up with more wounded creeps and fewer destroyed ones.

8.4.2.a Strategy T1: Focused Fire

FOCUSED FIRE was discussed earlier as a way of combating temporal attack window decay by more quickly destroying the lead creeps in a line. Two other reasons have been given for FOCUSED FIRE, one of which relates to converting wounded creeps to destroyed ones (the other is discussed in the next section).

FOCUSED FIRE places towers closely together so that the creep closest to one tower is closest to all of them and therefore all the towers attack the same creep. Although the strategy calls for a specific configuration of towers, the key issue is having all the towers attack the same creep. The impact of not doing so is trivially demonstrated in Figure 8.30 (left). Despite producing slightly more shots and doing more damage, no creeps were destroyed, resulting in a score of 0.

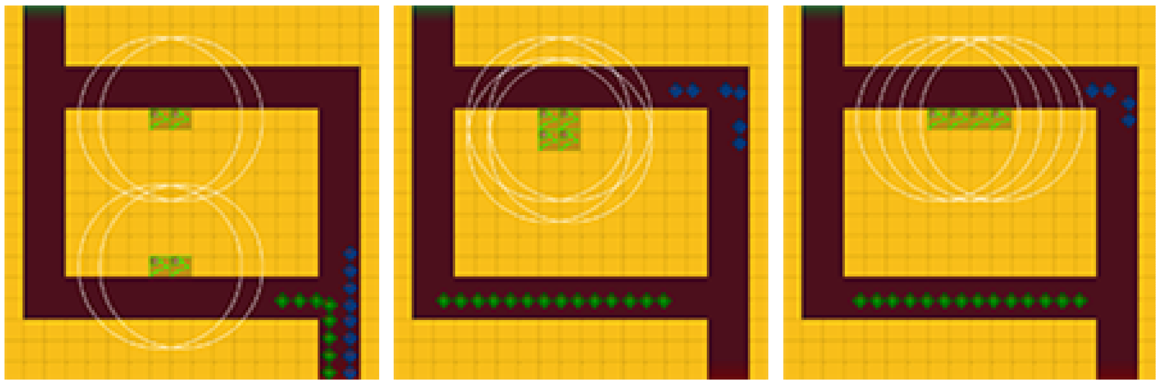


Figure 8.30. Effectiveness of Different Types of Focused Fire. Four level 10 Green3 fast attack towers. LEFT: Towers split between lines of creeps. Score=0. CENTER: FOCUSED FIRE. Score=8. RIGHT: Dense Chain. Score=10.

	Total Range	Number of Shots	Total Damage	Number of Kills
Split Between Lines	28	374	5,326,134	0
Focused Fire	24	362	5,069,796	8
Dense Chain	28	372	5,169,483	10

Table 8.3. Effectiveness of Different Types of Focused Fire. Scores are for strategies applied to Figure 8.30.

Figure 8.30 also shows how FOCUSED FIRE and Dense Chains performed. Dense Chains (right) did slightly less damage than the split lines approach (left) but destroyed 10 creeps vs. split lines' 0. Because FOCUSED FIRE (center) results in towers moving back from the path, it covers less range and therefore fired fewer times and did less damage. However, despite doing 5% less damage than split lines, it beats split lines' score of 0 by 8 points.

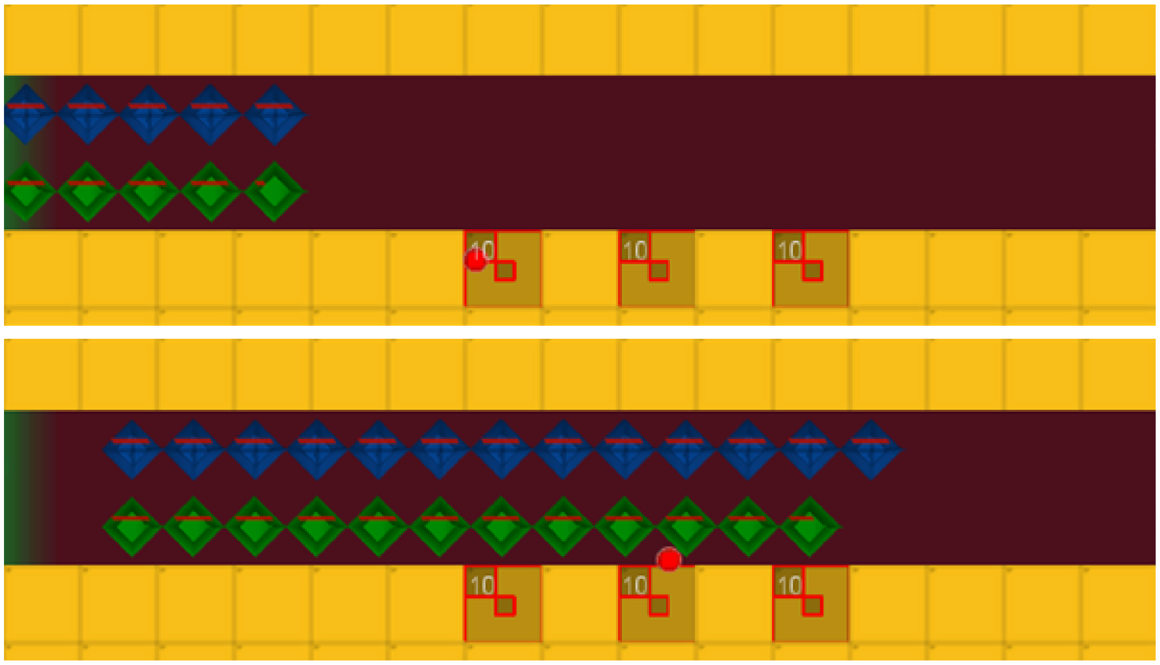


Figure 8.31. Effect of Tower Spread on Target Selection. TOP: Before entering the range, the closest creep for all towers is the bottom right creep. BOTTOM: After nine shots, creeps are directly above all towers. Each tower has a different closest creep and thus different target.

In all of the examples shown so far, all the towers in Dense Chain and FOCUSED FIRE selected the same towers, at least for the first few shots. In the previous section it was argued that the value of this was hard to see but that it might be dependent on the type of tower. The subject most explicit about the FOCUSED FIRE strategy used Red3 towers, which have a large range and a slow fire rate, making it more likely that creeps will be directly over the towers when the next shots are fired (when creeps are directly adjacent to the towers, each tower has a different creep that is closest to it; see Figure 8.31).

Figure 8.32 compares a dense chain and a slightly separated one when using level 10 Red3 towers. A level 10 Red3 tower can destroy a creep in three shots. In the dense chain (left), the towers always agree on the target. In the separated chain (right), the towers focus on the same target for the first five shots then select different targets. The dense chain destroys a creep every three shots while the separated chain, only slightly wider than the dense chain, destroys a creep every third shot for the first nine shots then distributes damage over different targets, not destroying another creep until seven shots later.

Although not shown in the image, the focused fire configuration earns a score of 7, directly between Dense Chain and the separated one. In all situations tested, Dense Chain outperforms focused fire. This might be because focused fire sacrifices usable range although it might also be because focused fire, despite the name, splits targets earlier than either the dense or separated chains. This is due to the bottom tower's position one tile below the path adjacent tiles increasing the effect of diagonal lengths to creeps at a distance. Despite the apparent superiority of Dense Chain, no subject ever explicitly stated that they used a Dense Chain because it reduced temporal attack window decay, increased conversion of wounds to kills or decreased overage.

8.4.3 To Reduce Overage

Damage is wasted when a shot does more damage than necessary. For example, if a tower's attack does 1,000 points of damage but the creep has been wounded to the point that it only has 10 health points left, 990 points of damage were wasted. This is called *overage*.

There are several strategies for dealing with overage such as using less powerful towers, using fast low power towers and intentionally leaking.

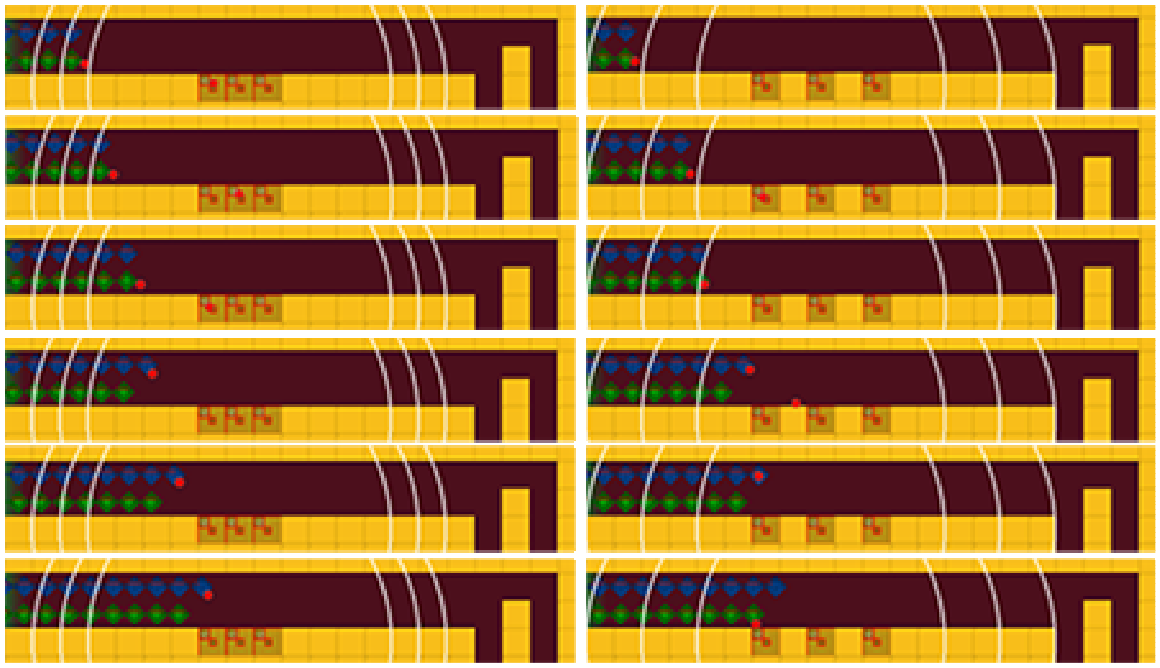


Figure 8.32. Dense Chain vs. Spaced Chain. Three level 10 Red3 slow, powerful attack towers can destroy a creep in three shots. LEFT: DENSE CHAIN. The first three shots target the same creep, destroying it. The next three shots target the same creep, destroying it. Final score=8. RIGHT: CHAIN. Similar to the layout on the right but with a single space between each tower. The first three shots target the same creep, destroying it. Shots four and five target the same creep but shot six targets a different creep. Two creeps are wounded but none are destroyed. Final score=6.

	Target Order	Shot Where Targets Split	Kill On Shots	Final Score
Dense Chain	1,1,1, 2,2,2, 3,3,3, 4,4,4	-	3, 6, 9, 12, 15, 18, 21, 24	8
Separated Chain	1,1,1, 2,2,3, 2,3,3, 5,9,17	6	3, 7, 9, 16, 20, 21	6
Focused Fire	1,1,1, 2,3,2, 2,3,3, 9,9,9	5	3,7, 9, 12, 16, 21, 22	7

Table 8.4. Target Selection for Focused Fire. Scores for the configurations shown in Figure 8.32. Target Order refers to which creep (from bottom right) is hit. Shot Where Targets Split is the first shot where the towers have fallen out of synchronization and selected different targets. Kill On Shots shows which shots destroyed a creep.

8.4.3.a Strategy T1: Focused Fire

Focused fire has been discussed extensively in the earlier two sections. One additional reason given for using focused fire is that it reduces overage. The reasoning goes like this: only the last shot (the one that kills the creep) has the possibility of overage, so if every tower attacks the same creep, only one of those towers risks overage. The others use their strength to the fullest. The alternative is that each tower attacks a different creep, each incurring overage on their last shot. Although we have not tested the effectiveness of this strategy, it is pointed out that the number of shots that risk overage is equal to the number of creeps destroyed, so if nine creeps are destroyed by three towers, three shots risk overage regardless of whether the towers are focused on one creep at a time or targeting different creeps.

8.4.4 To Combat Side Bias / Uneven Coverage

As one subject noted, a given position is always closer to one line than the other, at least for a given attack window. If all towers are closest to the same line, one line will be attacked more than the other, causing an uneven distribution of damage. In addition, because range boundaries are circular, the closest line goes through a thicker part of the range than the outer line, causing the outer line to be attacked for less time. On the other hand, around corners, U-turns and islands, the outer line takes a longer path, so perhaps it's the inner line that's in range for a shorter period of time (different subjects noticed different aspects of this problem and did not agree on whether it was the inner or outer line that was the biggest contributor to their losses).

Several subjects believed uneven coverage lowered their score by lowering their ability to destroy one of the two lines of creeps. We have not yet tested whether this belief is correct.

Strategy T2: Two-Sided Kill Zone

A TWO-SIDED KILL ZONE places towers on both sides of a path so that both lines are attacked.

In the example in Figure 8.33, the subject placed offense towers on corners to get the MAXIMUM USABLE RANGE. After placing them, they placed offense towers on the opposite side of the path so that both lines that passed would be attacked. The area opposite a corner is an interior corner, which only allows a tower to use 25% of its range, so it appears that the desire for dual line coverage and MAXIMUM USABLE RANGE work at cross purposes.

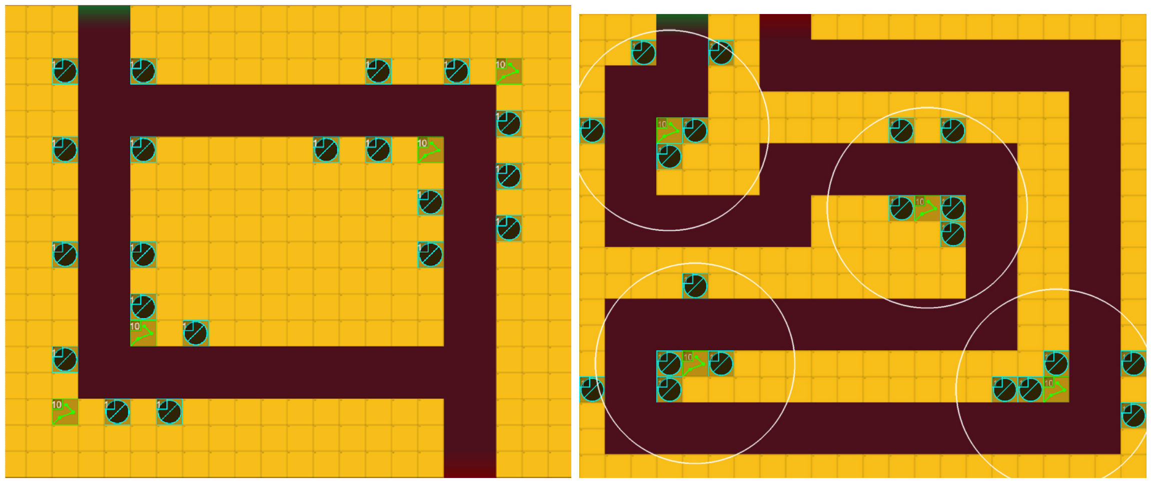


Figure 8.33. Strategies for Handling Side Bias. LEFT: TWO-SIDED KILL ZONE (3N0100). The subject places offense towers on the corners because they have the MAXIMUM USABLE RANGE. They also place offense towers directly across from them on the interior angle (which has the minimum usable range of a path adjacent tile) so that they can have a two-sided kill zone. RIGHT: SLOW SMALLEST TEMPORAL ATTACK WINDOW (3N0600). Although towers are placed on both sides to slow both lines, they place more slowing towers on the corners where the offense towers are because the inside line has a shorter path around a corner and thus is in range less time than the outer line.

Strategy T3: Slow Smallest Temporal Attack Window

On a straightaway, the line of creeps closest to the tower goes through a thicker part of the range circle than the other line. Around corners and U-turns, the line of creeps closest to the tower have a shorter path. Most subjects considered the inner line to spend less time in range. In the interest of balance, many subjects tried to counter this by placing additional slowing towers near the inner path.

Strategy T4: Slow Most-In-Range

Some subjects argued that slowing should be focused on those creeps that go through the largest part of the range as slowing would have the most effect. Slowing a creep that is in range for four seconds leads to it being in range for eight seconds, a gain of four seconds. Slowing a creep that is only in range for two seconds leads to it being in range for four seconds, a gain of only two seconds.

Strategy T5: Slow Least-In-Range

Some subjects argued that slowing should be focused on those creeps that go through the smallest part of the range as those creeps are already receiving less damage and are thus less likely to be destroyed.

8.4.4.a See Also

- **Strategy TP2: Swamp**. THE SWAMP strategy distributes the creeps across dozens of kill zones, with each focusing on a single creep.
- **Strategy Family:** DIFFERENTIAL SLOWING. DIFFERENTIAL SLOWING results in each line coming through a kill zone at different times, allowing the kill zone to give each line full coverage.
- **Strategy TP16:** KILL ZONE JUMPING. Subject 3n1000 described KILL ZONE JUMPING, which keeps one line at one kill zone and lets the other move to a second one, as "removing temptation" from the tower since there is only ever one line it can consider attacking. The kill zone can focus on one line and then the other, getting full coverage on both lines.

8.4.5 To Combat Target Thrashing

Earlier it was argued that it is better to do a lot of damage to one creep than a smaller amount of damage to a set of creeps. This requires the tower to focus on a given target as long as is possible. If the tower keeps switching targets, the damage is divided over a group of creeps, potentially preventing the tower from destroying the creeps and thus earning points.

Towers in GopherTD have sticky targeting – once a tower has selected a target, the tower attacks that target until it is out of range. Target thrashing often happens, however, because tower ranges are circular. This affects the large number of subjects who place towers on corners, U-turns or islands and use the Satisficing Strategy: One Back for Range.

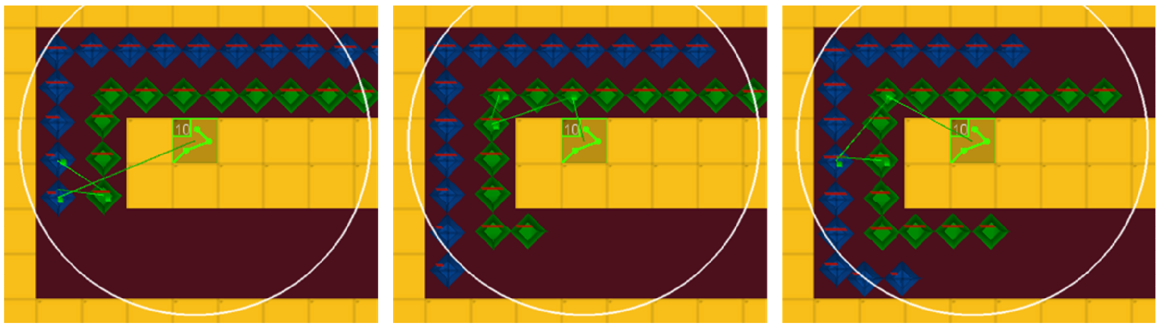


Figure 8.34. Uncovered Corners Cause Target Thrashing. LEFT: The tower is focused on the first blue creep (blue 1). CENTER: The targeted creep leaves range. The tower picks the creep closest to him, which is now green 8. RIGHT: The originally targeted creep re-enters the range. Despite being the original target and now heavily damaged, the tower stays with the second target.

In Figure 8.34, the subject has placed the tower one space back from the end of the U-turn, raising its usable range from 27 to 30. The tower has more range but no longer covers a small part of the bottom left corner. When the outer line of creeps moves through the corner, they briefly exit the tower's range, causing the tower to select a new target. When the creeps re-enter the range, they are no longer the closest creep and the tower has another target so these creeps, possibly heavily wounded, are able to continue through the tower's range with no danger of being attacked again.

It is worth mentioning that the map shown in Figure 8.34 is Elemental-ish and in the position shown the blue (outer) line is ahead by three tiles, causing them to enter the tower's range first. The path synchronization gap disappears as the creeps move around the corners but the tower's initial target was selected before then. Had the initial target been on the inside line, not covering the bottom left corner would likely not have been problematic.

Strategy T5: Cover Corners

Many problems, once identified, have trivially simple solutions. The solution in this case is to move the tower closer to the edge of the U-turn so that it covers the corner.

8.4.6 To Combat Selecting Wrong Target

In some Tower Defense games, the player can choose a targeting strategy for each tower. Options include nearest, strongest, weakest, fastest and, occasionally, the one closest to the exit. In some games, the player can spend money on bonuses and one-time use items, one of which allows the player to manually select a target.

Vector TD, the game upon which GopherTD was based, allows the player to select targeting strategies. We left this feature out for simplicity. Even so, players used a combination of positional and tower selection strategies to control which creep a tower targeted.

The two most common reasons for wanting to control targeting are to quickly eliminate creeps (see 8.4.1 To Combat Temporal Attack Window Decay) and prevent wounded creeps from making it to the exit. A common problem (and one that induced noticeable psychological distress in subjects) is that, given a choice between a weakened creep near the end of the tower's range about to make it to the map exit and a healthy creep closer to the tower, the tower will select the closer creep rather than the one about to exit the map (the player loses points for every creep that leaves the map). An example is shown in Figure 8.35.

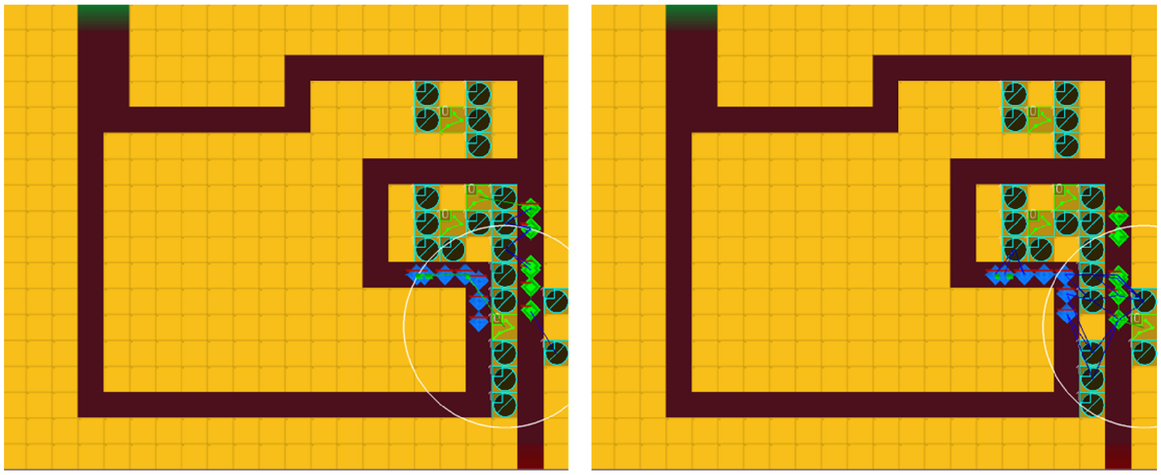


Figure 8.35. Strategy T7: Sacrifice Attack Window to Control Target Selection. LEFT: The green offense tower (white range circle showing) must choose between the healthy blue creeps on the left, that have not yet gone through the primary kill zone and the weakened green creeps on the right that have made it past the kill zone and are about to exit the map. The tower chooses the creeps on the left because they entered the range first. Score=24. RIGHT: STRATEGY T7: SACRIFICE ATTACK WINDOW. The green offense tower (white range circle showing) is moved to the outer wall. It covers significantly less useable area but is closer to the creeps exiting the map, causing it to prioritize attacking those. Score=26.

Strategy T6: Select Towers with Small Range

For a tower to select the wrong target, there must be more than one target in range. All other things being equal, the larger the tower's range, the more potential distractions there will be in range. One obvious solution is to use towers with smaller ranges. This gives the problem solver more control over which creeps are targeted. It also has other advantages. For a given type of tower, towers with smaller ranges (lower upgrade level) cost less and, for the same amount of money, have a higher damage potential (see Strategy E11: Use Many, Non-Upgraded Towers).

Strategy T7: Sacrifice Attack Window

Figure 8.35 shows a cleanup tower positioned near the exit. When placed on the left wall, it covers two areas, one where healthy creeps move before entering the map's primary kill zone and one next to the exit. These constitute two different spatial attack windows. Because the healthy creeps enter the attack window slightly before the wounded ones leaving the map, the tower focuses on the healthy creeps, ignoring the wounded ones that are about to leave the map and lower the problem solver's score.

In the right hand image, the tower has been moved from the left side of the exit to the right, an outer wall. The tower has significantly less usable range but is closest to the creeps exiting the map, causing it to focus on those creeps. This leads to stopping several wounded creeps, leading to a higher score despite reducing the usable range and number of attack opportunities.

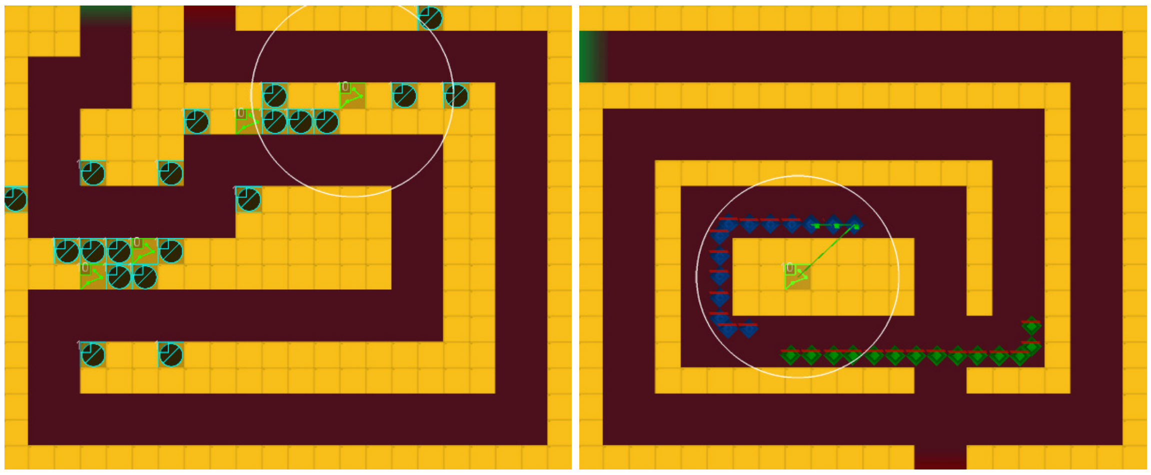


Figure 8.36. Strategies T8: Bias to Exit and T9: Avoid Islands. LEFT: BIAS TO EXIT. The green offense tower (white range circle showing) covers two areas but is placed so that it is closer to the side nearest the exit, causing it to prioritize creeps about to escape. RIGHT: Because of sticky targeting, towers focus on a specific target until it leaves range. Because the range of an island is so long, the tower does not select new targets often.

Strategy T8: Bias to Exit

Most of the walls are two tiles wide. Towers often cover two or more areas but, when wall thickness is an even number, the tower must be closer to one side than the other. To make sure the tower focuses on the creeps closest to the exit (or whichever area is most important), place the tower on the side closest to the more important area (see Figure 8.36).

Strategy T9: Avoid Islands

Towers in GopherTD use sticky targeting, which means that once they select a target (creep), they stick to that target until it leaves the tower's range. All other things being equal, the longer a continuous path through a range, the longer a tower stays on one target and the less often it switches to new targets. This allows one creep to essentially capture the towers, not allowing the tower to focus on other creeps, including those that are more important (e.g., ones near the exit).

8.5 Efficiency Strategies

Efficiency strategies try to optimize certain variables (e.g., money, range, power). There are three types of efficiency strategies.

- **Alternative**. Consider Strategy SP3: Maximum Usable Range – Traffic Volume. It says to place towers on those spots that have the highest usable range, regardless of where those spots are. The efficiency strategy Strategy E3: Concentrate to Share Slowing says to place offense towers near each other so that they can use the same set of slowing towers, thus saving money. This strategy is potentially mutually exclusive with the first strategy as the positions with the highest usable range might be on different corners of the map. While the two strategies can be combined through either prioritization (focus on one strategy, use the other to break ties) or compromise (rank each position based on a weighted combination of each strategies' criteria), it is non-trivial to do so.

- **Refinement**. Strategy TP1: Slow in Range says to place slowing towers where they will help offense towers by slowing the creeps in their range. Efficiency strategies such as Strategy E10: Slow at Entrance to Kill Zone (slow creeps as soon as they enter an offense tower's range) refine more general strategies by suggesting which of its existing options to focus on. These are normally easy to add to a set of strategies.
- **Constraint**. Strategy E1: Avoid Single-Sided Interior Corners says to avoid any space that is an interior corner as a tower there can only use 25% of its range circle. Unlike refinement strategies, constraints do not suggest an action, they only rule certain ones out. Like refinement strategies, these are normally easy to add to a set of strategies.

8.5.1 For range

Strategy E1: Avoid Single-Sided Interior Corners

A tower on an exterior corner can use approximately 75% of its range. A tower on a straight away can use 50%. Towers on an interior corner can use 25%. If a path moves past a position on two opposing sides, it is a double sided position. If a path moves past a corner on both sides, the position is both an interior and exterior corner. If a path moves past the corner a single time, it is a single sided corner. To maximize the amount of usable range, exterior corners make a lot of sense but might be precluded by other strategies the problem solver is using. Strategy E1: Avoid Single Sided Interior Corners can normally be used with any other strategy.

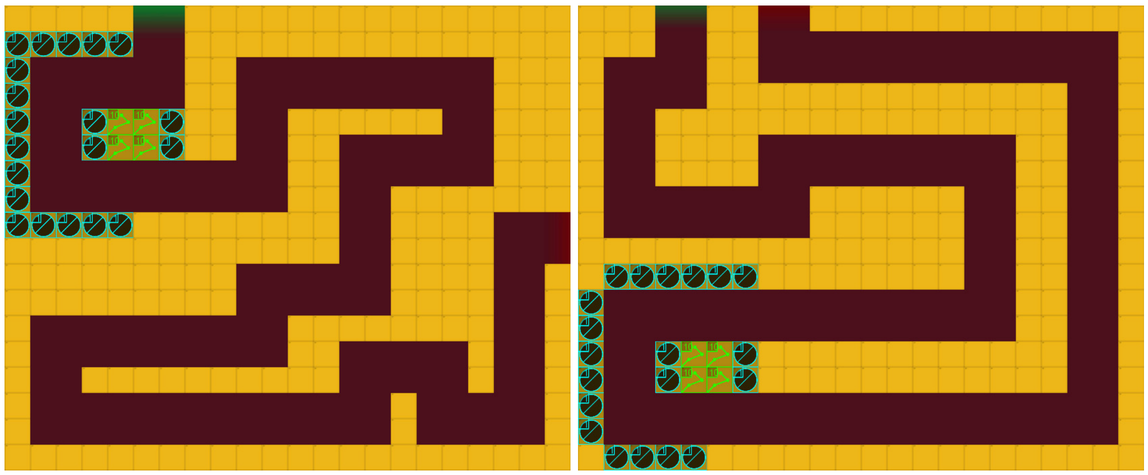


Figure 8.37. Strategy E1: Avoid Single-Sided Interior Corners. LEFT: Subject E0100 places towers around a U-Turn. The blue slowing towers form a chain along the outside of the U-Turn. RIGHT: Avoid Single Sided Interior Corners. Later in the testing process, subject E0100 uses the same strategy but, realizing that interior corners offer less usable range, avoids placing towers on the interior corners.

Strategy E2: Don't Upgrade Range into Walls

Many subjects know that upgrading towers upgrades their range but the usable range of the tower is only increased when the additional range covers additional path area. At some point, the new range crosses the path and starts covering the walls, which is unusable. Subjects who noticed (or at least remarked that they noticed) that the additional range went into the wall stopped upgrading when the additional range stopped helping.

8.5.2 For slowing towers

Strategy E3: Concentrate to Share Slowing

Strategy TP1: Slow in Range says you should use slowing towers to slow creeps when they are in an attack tower's range. When offense towers are spread apart, each needs its own set of slowing towers. If the offense towers are placed together, they can share slowing towers. In GopherTD phase three, this was less important as the subjects had to use 20 slowing towers, but when subjects are allowed to choose their own pieces, needing fewer slowing towers means the problem solver can purchase fewer slowing towers and spend the money on additional offense or some other capability.

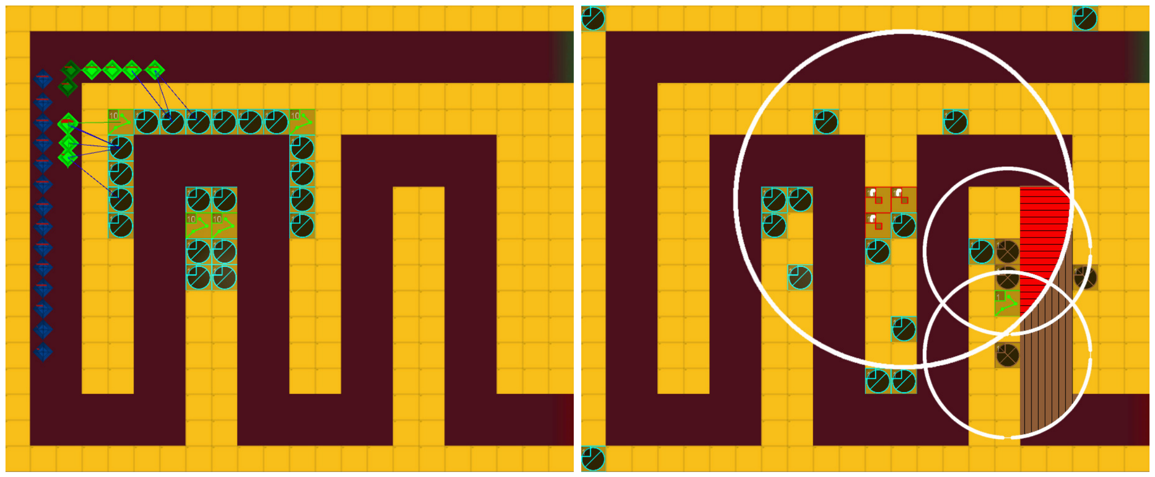


Figure 8.38. Towers with Multiple Roles. LEFT: RE-USE DIFFERENTIAL SLOWING PRE-KILL ZONE LIMITED VIEW SLOWING TOWERS (3E0400). The chain of blue slowing towers slows the green (inner) line of creeps, allowing the blue (outer) line to enter the kill zone alone. By the time the blue line enters the kill zone, the slowing towers are done delaying the green line and switch from DIFFERENTIAL SLOWING to SLOW IN RANGE, slowing the creeps just as they're attacked by the offense towers. RIGHT: RE-USE HEAVY AS CLEANUP (a0100). The level 10 Red3 towers in the center of the map have a large range (shown by large white circle). These heavy offense towers attack three areas in the center of the map. Near the end of the path is a cleanup area (two smaller white circles on the right side of the map) made up of four brown freezing towers and one level 1 Green3 fast attack tower to quickly eliminate weakened creeps. The heavy tower's range reaches the cleanup area (area shown in red, horizontally lined area), allowing it to lend significant (and significantly expensive) power to cleanup.

Strategy E4: Re-use Differential Slowing Pre-Kill Zone Limited View Slowing Towers

Strategy TP10: Pre-Kill Zone Limited View Differential Slowing says to place slowing towers where they can only see one line of creeps before entering the kill zone. It is okay for these slowing towers to see both lines after the creeps enter or pass the kill zone. On many maps, the slowing towers can be placed such that they play the role of DIFFERENTIAL SLOWING before the kill zone and, by placing them near the offense towers, switch to the role of Strategy TP1: Slow in Range once the creeps enter the kill zone. As in many of the efficiency strategies, using the same tower to play two roles frees up resources that can be allocated elsewhere.

8.5.3 For cleanup

Strategy E5: Re-Use Heavy as Cleanup

Heavy towers cost a lot of money and do a lot of damage. They are used to destroy healthy creeps. Sometimes a creep will make it past the core attack areas (known as *leaking*). Cleanup towers are weaker, cheaper towers whose job it is is to stop a small number of heavily weakened creeps. If a heavy tower has a large enough range, it can be placed such that part of that range covers the cleanup area, allowing it to switch from a powerful, expensive heavy attack tower to a powerful, expensive cleanup tower when conditions permit (i.e., when no creep in the primary kill zone is closer).

8.5.4 For limited good positions

On some maps, there are many good positions. On others, a few positions are significantly superior to the others on the desired criterion (usable range, TEMPORAL ATTACK WINDOW SEPARATION, path synchronization gap, etc.). When this occurs, the problem solver can place a few towers at the good locations but then must put the remainder of their towers at inferior positions, causing them to be less effective.

Strategy E6: Use Fewer, Stronger Towers

In GopherTD, where subjects received 50,000 credits and the cheapest offense tower cost 100, problem solvers could use up to 500 offense towers. If the subject chooses to use level three offense towers (the most powerful towers), the cheapest tower (Green3) costs 2,000 and the subject can use up to 25 offensive towers. If the most expensive tower is used, the 13,750 level 10 Red3 tower, the subject can only afford three towers. An obvious solution to having few good positions is to use fewer, stronger towers. Upgraded towers do less damage than spending the same amount of money on a set of non-upgraded towers (in terms of pure maximum damage capability, ignoring effects of individual and composite range; see Figure 8.42) but would be able to benefit more from a small number of superior map positions.

8.5.5 For limited good joint positions

One of the most common strategies, and one that was so obvious that few subjects deigned to mention it, is to place towers where they could attack both lines of creeps. On most maps this was not a problem, but on some maps there were relatively few places where a subject could place a tower such that it could attack both lines. One option is to place as many towers as possible at those positions, but those positions might be suboptimal in other regards.

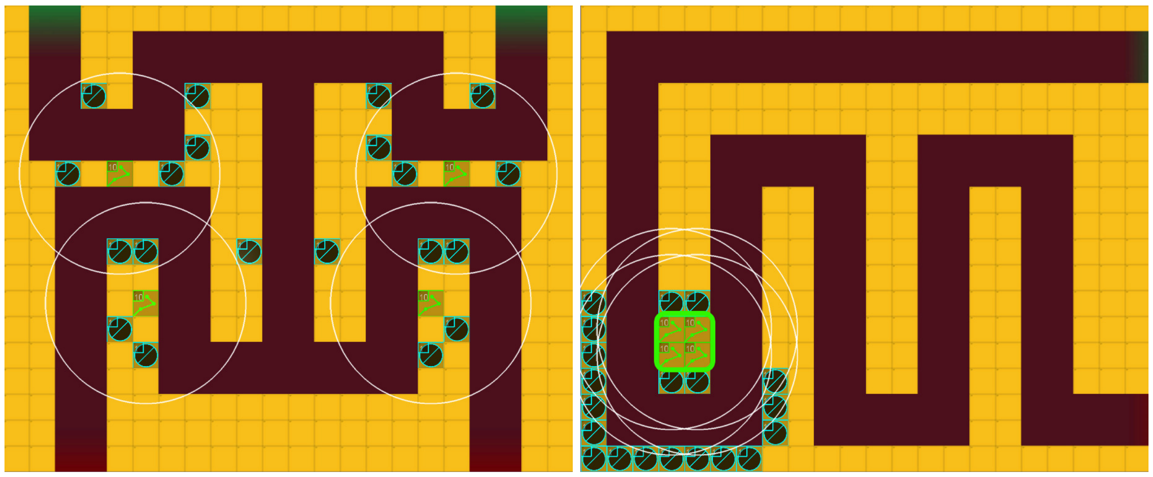


Figure 8.39. Strategies Strategy E7: Decompose Problem and Strategy E8: Create a Core of Multi-Target Towers. LEFT: Decompose Problem (3E1000). The creeps that come in the top left entrance are handled by the two offense towers on the left (range shown in white circles) while the right line of creeps is handled by the two offense towers on the right. RIGHT: Create a Core of Multi-Target Towers (E0100). Four Green3 offense towers (ranges shown in white, towers highlighted in square) are combined to create one meta-tower that can, in theory, attack 12 creeps at the same time.

Strategy E7: Decompose Problem

A few subjects chose to decompose the tower defense problem into two separate problems, each with one line of creeps. While not done on every map, these subjects did it on maps where the paths did not run side by side, even when there were several good positions for attacking both lines (see Figure 8.39, left). On spatially symmetric maps, the solutions were often symmetric and on spatially asymmetric maps, the solutions were sometimes quite different (different strategies could be used in each subproblem).

8.5.6 To use Area of Effect towers despite their not existing

GopherTD, unlike most tower defense games, does not have *area of effect* towers (see glossary). This was a disappointment to many subjects since this type of tower is the cornerstone of many subjects' strategies in other tower defense games.

Strategy E8: Create a Core of Multi-Target Towers

While GopherTD does not have area of effect towers, it does have Green3 towers, which attack three creeps concurrently. One subject (E0100) decided they could create a virtual area of effect tower by combining four Green3 towers into one tower that could, in theory, attack 12 (out of 28) creeps concurrently (Figure 8.39). While this at first appeared to be an attempt to misuse the available options to force the transfer of a favored but non-applicable strategy, this subject had one of the highest scores in the first phase of the experiment.

8.5.7 To fully utilize slowing

Strategy E9: Don't Slow Before Kill Zone

If the towers are attempting to execute Strategy TP1: Slow in Range, slowing creeps before they enter the offense tower's range can have some undesirable effects. First, the slowing is wasted – making creeps move slowly where they can't be attacked is not, in and of itself, helpful. Second, the part of the slowing tower's range that is outside of the offense tower's range could be moved inside the offense tower's range where it would be useful. Third, slowing towers are themselves slow – once they fire, they cannot attack again for two seconds. If they attack too early, they will not be useful for SLOW IN RANGE for a while. Finally, if EXPLOIT GEOMETRY or DIFFERENTIAL SLOWING has been used to separate the two lines of creeps, slowing the lead line before it is attacked gives the trailing line time to catch up, partially undoing the line separation.

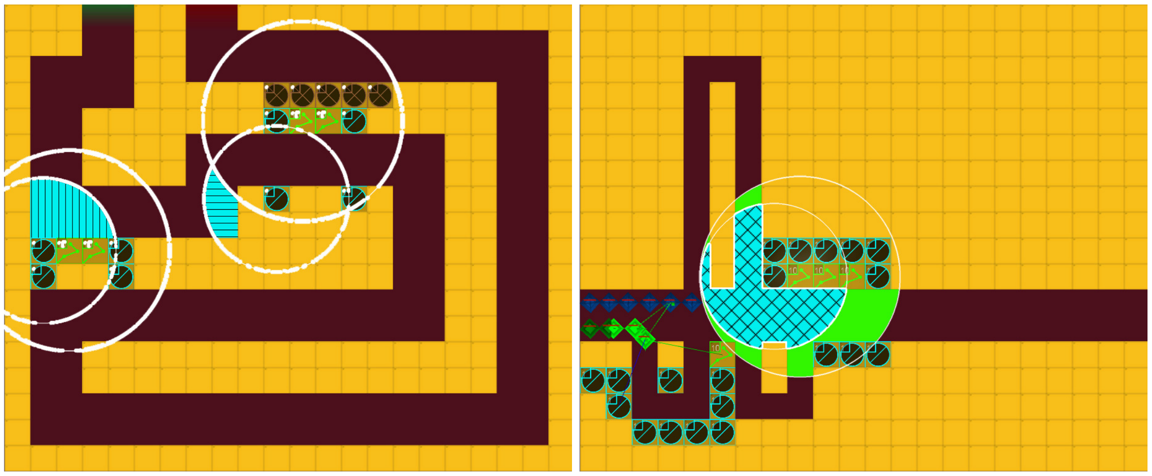


Figure 8.40. Range Alignment. LEFT: Poor range alignment (E1900). In the pair of towers on the left, the slowing tower (blue, shaded, vertically lined area) slows creeps after they have moved part of the way through the green offense tower's range (larger white circle). In the pair of towers on the right, the slowing tower (blue, shaded, horizontally striped area) slows the creeps before they enter the offense tower's range. RIGHT: Perfect range alignment (3E0900). The range entrance of the slowing tower (blue, shaded, cross hatched area) is perfectly aligned with the range entrance of the offense tower, causing creeps to be slowed the instant they enter the offense tower's range.

8.5.8 To fully utilize offense

Strategy E10: Slow at Entrance to Kill Zone

Strategy TP1: Slow in Range places slowing towers so that the creeps move slowly through an offense tower's range. The slowing effect lasts for two seconds. This means creeps might still be slow even after leaving a slowing tower's range, so it is okay if a set of slowing towers doesn't cover all of an offense tower's range. However, this uncovered space must be behind the slowing tower (as defined by path direction). If no slowing is applied at the offense tower's range entrance, the creeps will move full speed through part of the offense tower's range, cutting the number of attacks over that range in half (slowed creeps move at half speed and thus can be attacked twice as often). The solution is to slow creeps as soon as they enter a tower's range.

8.5.9 For power

Strategy E11: Use Many, Non-Upgraded Towers

All towers have a base cost and an upgrade cost. Upgrading a tower one level costs half the base cost, increases damage by almost half the base tower's damage and increases range by a small amount. For example, a base (level 1) Green3 tower costs 2,000, does 200 points of damage and has a range of 70. A level 2 Green3 costs 3,000 (+1,000), does 290 points of damage (+90) and has a range of 73 (+3). The numbers at level 3 are 4,000 (+1,000/2,000), 380 (+90/180) and 76 (+3/6).

When upgrading, cost increases faster than power, although not by much. The solution in Figure 8.41 (left) uses 20 offensive towers (18 green, two red) at a cost of 41,000. All towers are level one. This configuration is capable of an aggregate 61,000 points of damage per second. For comparison, we created a similar configuration using upgraded towers: three level 10 Green3, one level 2 Green3 and one level 3 Red3. Cost: 41,000. This configuration is capable of an aggregate 56,436 points of damage per second, a drop of 4,564 (-7.5%). For a fixed amount of resources, a set of many weak towers has a higher aggregate damage amount than a set of a few, upgraded towers.

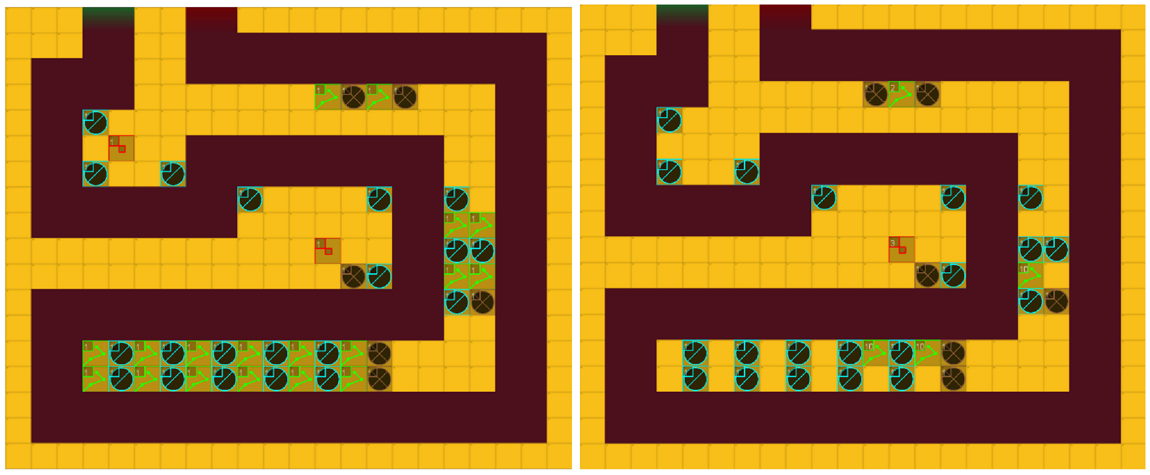


Figure 8.41. Strategy E12: Use Many, Non-Upgraded Towers. LEFT: Subject E1200 uses 18 fast offensive towers (green) and two powerful offensive towers (red), all level one. Cost: 41,000. Total Damage Per Second: 61,000. RIGHT: The same solution using four fast offensive towers and one powerful tower. Cost: 41,000. Total Damage Per Second: 56,436.

Subjects had access to the tower statistics (damage, range, firing rate, cost, etc.). Several subjects did calculations similar to those above. The calculations were given as the reason for why they followed this strategy.

It should be noted that this is a complex domain and cannot be fully captured by a measure such as aggregate damage over time. Actual damage done is a function of a tower's range, temporal attack window size, target selection and other factors. A solution that is capable of a higher aggregate amount of damage does not necessarily do more damage or destroy more creeps. In Figure 8.41, the solution with 20 offensive towers (left) does 10,573,319 points of damage, destroying 23 creeps. The solution with five upgraded towers (right) does 11,490,696 points of damage (+917,377, +8.68%) and destroys all 28 creeps. It is not immediately clear why this happened. One possibility is that the upgraded towers also had more range, although in aggregate the few upgraded towers covered much less range than the many, weaker towers (see Figure 8.42).

8.5.10 For range

Strategy E12: Use Many, Non-Upgraded Towers

As noted in Strategy E11: Use Many, Non-Upgraded Towers, upgraded towers have significantly more power and slightly more range. If total range is a priority, it should seem obvious that two towers covers more range than a single tower of the same type. While upgraded towers cover more range, the increase is small compared to adding a second tower. For example, a level one Green3 tower costs 2,000 and has a range of 70 and area of 15,394. A level 10 Green3 costs 11,000 and has a range of 97 and an area of 29,559, slightly less than the combined area of two level one Green3 towers. A level 10 tower costs as much as 5.5 level one towers (this is true for all towers). For the price of one level 10 Green3, one can buy five level one Green3 towers with a combined area of 76,969, 2.6 times the area of the level 10 tower.

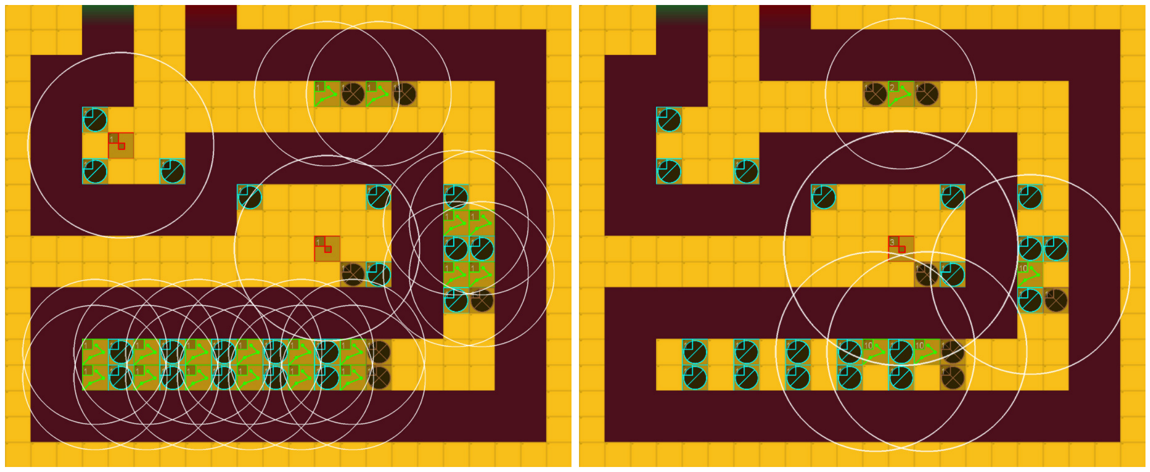


Figure 8.42. Base vs. Upgraded Towers. LEFT: Use Many, Non-Upgraded Towers (E1200). Range shown for 18 fast offensive towers (green) and two powerful offensive towers (red), all level one. Cost: 41,000. RIGHT: Range shown for 4 fast offensive towers (green) and one powerful offensive tower (red). Cost: 41,000.

There are caveats to these numbers. Splitting area over multiple towers allows the area to be distributed to multiple parts of the map. It is also less likely to have the end of its range going into walls. On the other hand, the center of the range is at the tower and the tower must be in a wall, so each tower will lose a large part of its range at its base.

8.5.11 For overage

Towers do a fixed, known amount of damage per shot. This damage is subtracted from the health of the target. If the damage is more than the health of the target, the target's health is reduced to 0 and the target is destroyed. The amount of damage done above what is necessary is called *overage*. It represents waste. Put more prosaically, it makes sense to use a rocket launcher against a tank but it is overkill when used on a small, injured bunny.

Tower	**Level**	**Cost**	**Damage Per Shot**	**Shots per Second**	**Type**
Green1	1	100	22	10	Fast
Green1	10	550	112	10	Fast
Green3	1	2,000	200	10	Fast
Green3	10	11,000	1,010	10	Fast
Red2	1	800	600	10	Fast
Red2	10	4,400	3,048	10	Fast
PurplE1	1	300	2,650	0.5	Strong
PurplE1	10	1,650	13,486	0.5	Strong
Purple3	1	2,800	22,000	0.5	Strong, Slowing
Purple3	10	15,400	112,000	0.5	Strong, Slowing
Red3	1	2,500	30,000	0.5	Strong
Red3	10	13,750	152,724	0.5	Strong

Table 8.5. Offense Towers Statistics. There are three colors, three types of towers per color and 10 power levels per tower type for a total of 90 types of offense towers.

There are several types of offense towers in GopherTD. They vary in power, speed and cost. Examples are shown in Table 8.5. The weakest tower (level 1 Green1) does 22 damage per shot while the strongest (level 10 Red3) does 152,724. The latter's shot is expensive not just in tower cost (13,750 vs. 100) but time – it fires only once every two seconds.

There are two problems with overage. The first is efficiency. The creeps used in the experiments have 370,012 points of health. A level 10 Red3 tower can destroy one creep in three shots. For the first two shots, the tower does full damage (152,724 per shot). After these, the creep has 64,564 points of health. The final shot exceeds this amount by 88,160 points, wasting 58% of its power. In aggregate, 28 creeps have a combined health of 10,360,336 and require 84 shots doing 12,828,816 points of damage to be destroyed. The aggregate overage is 2,468,480, a 19% loss. The amount of power wasted is enough to destroy 6 creeps, 21% of the total number. If the damage were distributed perfectly with no overage, the 28 creeps could be destroyed in 19% fewer shots (68 vs. 84), meaning the scenario could be won with 19% tower active time. Since aggregate active time (as a function of damage per shot) is the primary constraint on success, this is a significant loss.

The second issue with overage is opportunity cost. Damage wasted in overage on one creep is damage that could have been applied to another creep but was not. An inefficient attack on one creep precludes a more efficient attack on another and, assuming time is a fixed quantity (which is true for a given configuration of towers on a map but not for the map itself, as discussed in other strategies), might result in fewer creeps being destroyed.

Strategy E13: Use Low Power Towers

The magnitude of overage is related to the amount of power done by the tower. If lower power towers are used, it will take more shots to destroy the creep but the final shot will have less overage, leading to a higher overall efficiency. For example, three level 10 Red3 towers cost 41,250 and do an aggregate 229,086 points of damage per second. The same amount will purchase 16 level 1 Red3 towers that do an aggregate 240,000 points of damage. The waste due to overage for the level 10 towers is 19% vs. 5% for the level 1 towers.

Strategy E14: Use Fast Towers

The negative impact of overage is related to the number of attack opportunities lost. For example, consider a level 10 Red3 tower that has two creeps near the end of its range (technically, near the exit of its last spatial attack window). One creep has 217,000 points of health, the other has 100 points. If the tower destroys the one with only 100 points of health, it wastes 99.9% of the shot's power and cannot fire again for two seconds, enough time for the second creep to move out of range.

A level 10 Green3 does much less damage (1,010 vs. 152,724) but fires 20 times as often. The red tower has an advantage in damage per second (20,200 vs. 76,362) and damage per second per cost (0.92 vs. 5.55) but in the above example, within a two second period of time, the green tower would have destroyed the critically weakened creep and attacked the second one 19 times.

Strategy E15: Use Supporting Offense

In phases one and two, subjects could use any combination of towers they chose. Some subjects chose to use strong, slow, expensive towers to cause the primary damage to the creeps and a set of cheap, weak towers to finish them off.

Some subjects used lower power versions of the slow towers (i.e., the support towers were also slow). In the case of one powerful tower and four support towers, each with the same rate of fire, there is an 80% chance that a weak tower will destroy a critically weakened creep (all other things being equal), thus lowering the waste due to overage.

Other subjects used fast towers, which fire 20 times as fast as the slow towers. At first, it might seem like this would guarantee that the fast towers finished off the weakened creeps but even the most powerful of these does not do a significant amount of damage. Consider a slow level 10 Red3 tower (slow) and a fast level 10 Green3 tower. The red tower does 152,724 points of damage per shot. The green tower does 20,200 points of damage in the two second period between the red tower's shots. When facing a single healthy creep, the red tower's first shot reduces the creep's health to 217,288. The green tower reduces this to 197,088. The red tower's second shot reduces the creep's health to 44,364. The green tower reduces this to 24,164. The red tower then destroys the creep on its third shot.

A level 10 Red3 can destroy a healthy creep in three shots with no assistance. In this scenario, all of the damage done by the level 10 Green3 tower is irrelevant. Every point of damage it did is reflected in the increased overage on the red tower's final shot. Because the green tower costs 22% of the problem solver's total budget, this is an issue far more important than the overage issue.

In contrast, if the problem solver had purchased two of these towers, the green towers would destroy the creeps, significantly reducing the total amount of overage and enabling the red tower to destroy creeps in two shots, increasing the red tower's effective power by 33%. Because level 10 Green3 towers are only 20% less expensive than level 10 Red3 towers, the purchase of the green towers precludes the purchase of a similar number of red towers. We have not done the analysis to determine under which conditions, if any, this strategy is beneficial.

Strategy E16: Purposely Leak

Strategy P1: Use Cleanup Towers says to place towers near the map exit to catch those creeps that make it past the primary kill zones. These towers are generally not strong enough to destroy a healthy creep (if they were, they would be a primary kill zone, not cleanup) but can finish off a small number of significantly wounded creeps or a large number of creeps near death.

One option is to place the strong offense towers such that they cannot destroy the creeps, only significantly wound them. These creeps would be allowed to leak under the assumption that the cleanup towers would deliver the finishing blow. Because these are normally weaker towers, the overage from these towers would be smaller.

Strategy E17: Clean Between Attack Windows

Similar to Strategy E16: Purposely Leak, the attack window of a kill zone could be designed to significantly wound but not kill some number of creeps. In the space between when the creeps exit one of the kill zone's attack windows and enters the next would be a set of weaker towers whose job it is is to destroy the severely wounded creeps. This differs from Strategy E15: Use Supporting Offense in that the towers are placed in between a kill zone's attack windows rather than inside them. It differs from Strategy E16: Purposely Leak in that the cleanup towers designed to handle leakage are placed after all of the primary kill zones. Cleanup zones are also typically designed to handle a small number of creeps while this is not necessarily true of the others.

8.6 Polish Strategies

Polish strategies are supplementary strategies used to shore up problems with the primary strategies or otherwise improve the overall solution.

8.6.1 To handle leakage

If a kill zone is designed to stop all creeps and a few make it through, it is said to have leaked creeps. If there are no other towers to stop them, these creeps will make it to the map exit, lowering the problem solver's score.

In addition to lowering the problem solver's score, it was observed in all three phases of the experiment that leakage caused noticeable psychological distress in a large number of subjects. As the game is deterministic and full information, when a creep had made it past all the towers, it was clear to the subject that the creep would escape. Once the wave began (i.e., the subject was done building and the creeps started to move through the map), subjects could not directly intervene and thus lacked any control over the situation.

They could only watch as the creeps moved through the maze and eventually exited it, ending the game.

Strategy P1: Use Cleanup Towers

A perfect solution does not leak. When a solution is designed relatively well, the leakage should be only a few creeps and they should be heavily wounded. These creeps can be dealt with by placing a few weak, inexpensive towers near the map exit whose sole job is to stop leaked creeps. This strategy was not universally used but was quite common in the first two phases. Because subjects were assigned towers in phase three and did not have access to inexpensive offense towers, few subjects in phase three created cleanup areas.

Cleanup towers are typically weak as the creeps are assumed to be near death. Most subjects chose fast towers, although this might be because, in GopherTD, the cheapest tower is a fast tower. Almost all subjects used at least one slowing tower and one offense tower.

Strategy P2: Use Freezing For Cleanup

GopherTD has two types of slowing towers. Blue towers target four creeps simultaneously and cause them to move at half speed for two seconds. The more expensive brown freezing towers cause a single creep to stop completely for two seconds. Both towers fire once per second.

The majority of subjects concluded that brown freezing towers were ineffective as they stopped a single creep and then didn't fire again for two seconds, enough time to allow most of the creeps to move through its area (because of sticky targeting, even when the tower was able to fire again, it attacked the same creep as before, allowing all of the other creeps to move through its range unharmed). While a small number of freezing towers do not work well with large groups of creeps, there should be very few creeps that make it to the cleanup area. In these areas, one freezing tower can stop a creep long enough for even a weak offense tower to do a fair amount of damage. Two freezing towers can stop a single creep for quite a long time, allowing weak offense cleanup towers to be effective even against moderately healthy creeps. Three freezing towers can stop a single creep permanently.

Strategy P3: Relocate Towers

A few subjects noticed the leakage and attempted to solve the problem by moving one of the primary attack towers from the primary kill zone to the map exit. The powerful cleanup tower was capable of destroying severely weakened creeps but, because they had taken the tower from the area where it had been making the creeps severely weakened, the creeps were no longer weak when they reached the exit. By moving a strong attack tower from a position originally chosen for its tactical value to a position chosen primarily for its proximity to the end of the map, the overall solution was generally weakened and the subject's score dropped.

Strategy P4: Build at the Exit

Several subjects placed their primary kill zone next to the map exit. This in no way improved the quality of the solution nor was it intended to. The subject's primary concern was the amount of time they had to watch the leaked creeps move through the map and the psychological distress caused by that. Moving the kill zone to the map exit decreased the duration of this event.

8.6.1.a See Also

Two previously discussed strategies also address this problem.

STRATEGY T5: COVER CORNERS

STRATEGY E5: RE-USE HEAVY AS CLEANUP

8.6.2 To make pretty

Strategy P5: Place Towers Symmetrically

Many, though not all, subjects placed towers or tower groups symmetrically. On maps where the geometry was mirrored across the center point (as in Figure 8.43), the whole solution would be symmetric. On asymmetric maps, towers were placed such that they were locally symmetric. Some subjects volunteered that they were placing towers symmetrically. Others explicitly gave symmetry as an answer when asked why they placed the towers where they did. Several subjects said they used some functional strategy to place the first towers and symmetry to place the later ones "because it makes it look pretty". No subject stated that they believed symmetry helped their score. Symmetrical placements might help participants to remember their prior placements.

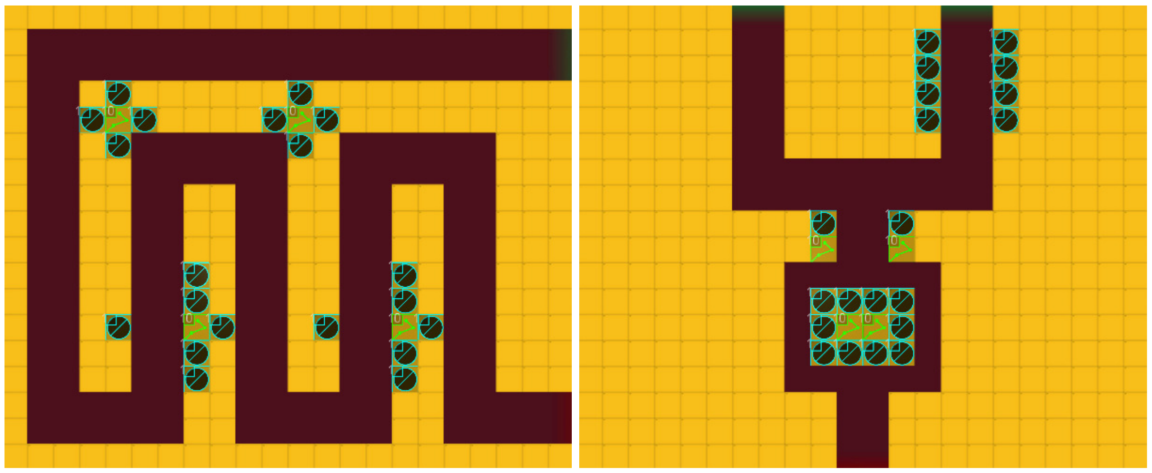


Figure 8.43. Tower Symmetry. LEFT: PLACE TOWERS SYMMETRICALLY (3E0600). Asymmetric map, locally symmetric kill zones. Four kill zones, two consisting of five towers, two with seven. RIGHT: PLACE TOWERS SYMMETRICALLY (s0400). The towers in the kill zone (map bottom) are symmetric but the right path (top right) has DIFFERENTIAL SLOWING towers not present on the left path.

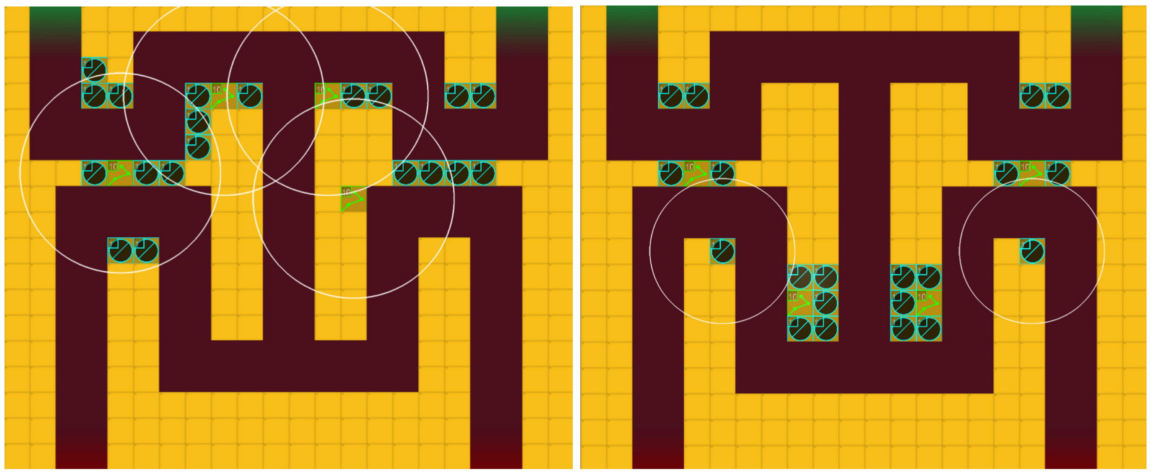


Figure 8.44. Symmetry Errors. LEFT: (3E0500). An example of an unnaturally asymmetric solution (none of the offensive towers are placed in similar positions). The asymmetry does not result from a strategy such as DIFFERENTIAL SLOWING that requires asymmetry. RIGHT: PLACE TOWERS SYMMETRICALLY (3N0700). Subject makes a common error in placing two towers (in white circles) at spots that they believe, incorrectly, are symmetrical.

8.7 Other Observed Strategies

8.7.1 To Improve

In all three phases, subjects had a training and a testing phase. Our goal was to get subjects to response stability so that we would have several data points (map solutions) from which to infer strategies and representations. Several subjects were fairly stable in responding, a few had no consistency from map to map and many were relatively stable except for small things that varied from map to map, either in the set of strategies they used or the execution of one of the strategies. When asked about changes during the testing phase, subjects often said that they were trying new things to see if other possible approaches worked better. The goal appeared to be exploring options for the purposes of improving their performance.

Strategy O1: Incremental Change

Many subjects, perhaps most, varied their strategies slightly from map to map. When asked why, on a specific map, their strategies changed, the subject would often say something such as "I was wondering if it would be better if I placed the slowing towers on both sides of the path rather than one." This was often followed by a conclusion such as "It didn't seem to help so I didn't do it on the later maps" or "I think I ended up doing better than I would have using my normal strategy so I kept doing this on the next maps." When a change worked, the subject would sometimes explain why they believed it helped, other times they would say they weren't sure. Whether or not an explanation was offered depended more on the subject than on the change or map – some subjects always tried to explain a phenomenon, others simply accepted that it did or did not work.

8.7.2 To Fight Fairly

Strategy O2: No Sneak Attacks

In an unexpected display of ethics towards the creeps, which are colored diamonds devoid of personality or individually distinguishing features, subject 3N0700 refused to place towers near the map entrance because "ambushing them at the door seems mean." It is not clear whether this is internally derived (i.e., it feels wrong to them to take this action) or external (i.e., it feels wrong to take this action towards this target), nor is it clear whether the subject is anthropomorphizing the creeps, although getting emotional towards and attributing intentions and feelings to the creeps was common among subjects.

Similar to STRATEGY P5: PLACE TOWERS SYMMETRICALLY, this bias is not stated in functional terms and the subject seems aware that it will not contribute to their goal of solving the puzzle. It is possible that biases such as these lead to better outcomes in similar problems (e.g., dealing with a living enemy) and that the bias is demonstrated here as a consequence of analogical transfer from other problems to this domain. Since the biases identified here have minimal impact on the subject's performance (assuming there are equally good positions away from the map entrance), there is assumedly little need to expend the mental resources necessary to reconsider and remove these biases.

8.7.3 Atheoretical

Most subjects explained their actions in the form of "I noticed problem/opportunity X so I decided to take action Y." A sizable minority explained their actions without resorting to identifying problems, opportunities or features or coming up with theories about why an action should work.

Strategy O3: Empirical Evidence

A few subjects explained their actions by stating "I don't know why this works but I tried it in training and it seems to work". When asked about these in the after-action review, some subjects tried to guess why it might have worked (at least one revised their explanation while discussing it), others did not. The number of subjects who did this was fairly small.

Strategy O4: Recall

In phases one and two, subjects trained on a subset of the maps, then tested on those maps (to establish a baseline) plus new ones. In phase three, subjects trained on all maps; there were no new maps in testing. One of the consequences of this was that many subjects attempted to recall their solution from training rather than use the strategies they had learned to re-create them. Many of the solutions began with "let me see if I can remember how I did this in training". It is possible that they were attempting to recall their earlier solution to think about it rather than replicate it. We did not create a method for capturing this information.

Subjects often believed that their score in training was higher than it actually was and often blamed low scores in testing on their inability to remember the specific tower layout they used in training.

Subjects offered explanations for their solutions but it is not clear how much of those explanations were current thoughts and how much were an attempt to recall their thought process during training.

Strategy O5: Mental Simulation as Measurement

Spatial strategies have the advantage of leveraging innate human capabilities. Precisely measuring time is, for humans, more difficult. For subjects that explicitly focused on time, one solution was to mentally simulate the scenario and use that to convert time into spatial measurements. As described by one subject:

> "And then I try to bring the blue towers in to try to slow down ahead of time… And the whole time I play I'm usually trying to figure out what exactly two seconds is in terms of squares, so, I try to anticipate that a little bit but… sometimes I'm off because I can't figure it out… I'm a horrible judge of what two seconds is. That really helps in this 'cause if you can figure that out perfectly in terms of like a grid you know exactly where to place everything." (3E0500)

8.7.4 Problem Unknown

Most subjects explained their actions in the form of "I noticed problem/opportunity X so I decided to take action Y." A few explained their actions in the form of "I decided to take action Y because it just seems to make sense." Unlike the strategies mentioned in the other sections, no rationale was given for why the strategy should work nor which problems they were attempting to solve. As such, the discussion of these strategies is necessarily brief.

Strategy O6: Keep Slow At All Times

The subjects who used this strategy believed that the creeps should always be slowed. Slowing was not limited to areas where the creeps were in range of an offensive tower. The rationale appears to be that if a little slowing is good, more slowing is better. An example is shown in Figure 8.45.

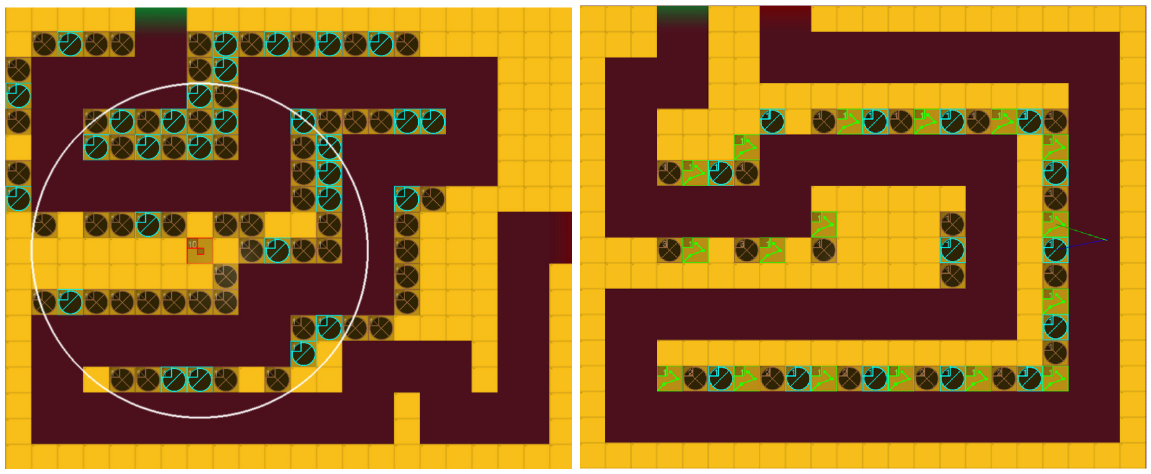


Figure 8.45. Strategies for Constantly Acting on Creeps. LEFT: KEEP SLOW AT ALL TIMES (e2500). A mix of Blue slowing and Brown freezing towers keep the creeps slow or frozen at all times. The offense is a single level 10 Red3 tower (white circle). RIGHT: KEEP ATTACKED AT ALL TIMES (e2100). The subject formed a chain of alternating slowing and offensive towers. The solution uses 14 Blue slowing towers, 19 Brown freezing towers and 17 Green3 fast attack towers. All Blue and Green towers and most Brown towers are level 1.

Strategy O7: Keep Attacked At All Times

The subjects who used this strategy believed that the creeps should always be attacked. Offense towers were placed so that there was no time during which the creeps were not being attacked by someone. The rationale appears to be that if a few attacks are good, more are better. Because subjects did not have enough money to buy powerful offense towers for every location on the map, to accomplish this strategy the subjects were required to use a large number of weak towers. An example is shown in Figure 8.45.

Strategy O8: Spread Out Offense Towers

Several subjects believed that the offense towers should be spread out, preferably such that none of their ranges overlapped. While no justification was given for this approach, subjects acted as if only one tower was allowed to attack at a time or that two towers attacking did the same amount of damage as a single tower.

Strategy O9: Join Split Lines

In STRATEGY TP8: EXPLOIT GEOMETRY and STRATEGY FAMILY: DIFFERENTIAL SLOWING, subjects noticed that one line of creeps would sometimes fall behind the other and formed strategies around this, attempting to maximize this property (path synchronization gap). At least one subject recognized this effect/feature but believed it was damaging and attempted to undo it, slowing the line that was ahead so that it would fall back in line with the other one. The subject felt the property was important but assumed it was a problem rather than an opportunity.

Strategy O10: Soften Targets

Before the creeps make it to the primary kill zones, weaker towers can attack them to "soften them up". It is not clear whether this helps or why it is done and subjects did not explain why this should be beneficial. The author of this paper used this strategy at the start of the GopherTD project and still does not know why; it just seemed like a smart thing to do.

8.8 Relevance to Other Tower Defense Games

Many of the strategies listed here seek to maximize a desirable criterion such as path synchronization gap, TEMPORAL ATTACK WINDOW SEPARATION or, more fundamentally, tower active time. Despite this, these criteria are not inherently good nor intrinsically beneficial. Their value is determined by other factors of the game such as tower capabilities and creep behavior. Under other circumstances, the strategies described above might not just be ineffective, they might be actively harmful.

8.8.1 Waves

The most obvious difference between GopherTD and other Tower Defense games is that GopherTD is not wave-based. In most Tower Defense games, the player starts with only enough money to buy a few low-powered towers. They receive money for every creep they stop which is used to buy more and more powerful towers. After a wave (set) of creeps is destroyed, the player is given a short amount of time to buy, sell (at a loss), move and upgrade towers. After this time passes, a new wave of stronger creeps arrives. In many games, the player earns interest on any money unspent at the end of a wave. These circumstances often cause different behaviors. It is common to buy just enough towers to stop a single wave. Players place towers at non-optimal positions early in the game to save the best spots for the more expensive towers. Players switch strategies as more money and tougher creeps cause the player to move to a different set of towers with different affordances. This is especially common when a superior strategy only works with a tower that can only be afforded later in the game.

8.8.2 Towers

The second most obvious difference between GopherTD and other Tower Defense games is the type and variety of towers. The 11 types of towers in GopherTD fit into the three categories slowing, slow and powerful and fast and weak. Two other tower types, Area of Effect and Damage Over Time, exist in most other Tower Defense games.

Area of Effect towers (sometimes called splash damage towers) affect every tower in their range. These towers are normally slow. The related strategy is to group all the creeps together so that the tower can hurt all the creeps in one hit. This leads to the player attempting to minimize the gap between creeps, increase the spatial attack window density and possibly ignore TEMPORAL ATTACK WINDOW SEPARATION, the opposite of what is sought by most of the strategies above.

Damage Over Time (DOT) towers do a small amount of damage that repeats for a period of time. Common instantiations include fire and poison. In some games, Damage Over Time inoculates – while a creep is being damaged by that damage type, they cannot be hurt again. Concretely, if a poison tower poisons a creep for 10 points of damage per second for five seconds, other poison towers cannot poison that creep during that period of time, even if the later towers do more damage. Under these circumstances, the obvious strategy is to spread out the Damage Over Time towers. In some games, DOT damage stacks (i.e., it can be poisoned multiple times concurrently). Either way, it is often undesirable to place Damage Over Time towers before slow, strong towers in order to minimize overage (i.e., if a tower does 100,000 points of damage and poison towers have weakened a creep to 10 points of health, the powerful tower will waste the majority of its power and be unable to fire again for a while). DOT towers work well for cleanup but should not be placed too close to exits as there will not be enough time to take advantage of their abilities. Players are less likely to use slowing with DOT towers as the creeps continue to take damage even after they've moved out of range.

Slowing works differently in many games. Many slowing towers do no damage, changing or eliminating the Swamp strategy. Many slowing towers increase their abilities as they are upgraded (i.e., a level one tower might slow 5% while a level 10 tower slows 75%). In GopherTD, slowing lasts for two seconds but can last much longer in other games. Combined, these two facts make a few upgraded slowing towers preferable to many non-upgraded ones as in GopherTD. Slowing towers are often more expensive in other games, making one less likely to use them for strategies such as DIFFERENTIAL SLOWING (if the player can only afford one, it is probably better used for the SLOW IN RANGE strategy).

In GopherTD, upgrades are symmetric and linear. Every upgrade level for a given type of tower costs the same and gives the same increase to power and range. This is not true in many Tower Defense games. It is common for a very cheap tower to gain little when upgraded from level one to two but to become the most powerful tower in the game when fully upgraded. In many cases, the type of tower changes. For example, in Desktop Tower Defense, the cheapest tower is the pellet tower, considered almost worthless. However, when fully upgraded, it becomes a powerful sniper tower that targets the strongest creep on the map, each shot reducing its health by 10%, regardless of how strong the creep is.

In some Tower Defense games, you can choose which attributes of a tower (power, range and speed) to upgrade. In others, you can add bonuses to towers. In Ghost Hacker, you can buy power-ups to give towers the ability to do area of effect damage, damage over time damage, slow creeps, etc. In VectorTD you can earn power-up "towers" that cause all towers in their range to do more damage or cover more area. This often causes players to cluster their towers and reserve positions in the center of a cluster for the upgrade tower.

In some games, towers have special abilities. In Onslaught, cannons periodically go into "freakout" mode, during which they shoot significantly faster. Onslaught also has tower combos. If two or more towers are close to each other, they can help each other. For example, if a rocket (launcher) and laser are placed next to each other, both have their damage fully upgraded and if the creeps enter the range of the rocket first, the rocket launcher will launch missiles that circle around the towers and fire lasers at the creeps. Other combos include napalm rockets (two rockets, one cannon and a taser), a black hole (three tasers and a laser) and a laser landmine (two cannons, two rockets and a laser).

Towers sometimes vary because creeps vary. GopherTD is based on Vector TD. While we used most of the design of Vector TD, we did not include the concept of color. In

Vector TD, creeps had colors just like towers. When the creeps were the same color as a tower, the tower did 50% more damage. When it had the designated opposing color (i.e., green was the opposite of red), the tower did half damage. In Vector TD, this most likely led players to have a few towers of each color, leading to more tower diversity than we saw in GopherTD.

8.8.3 Creeps

Three types of creeps GopherTD didn't have were fast, slow-resistant and flying creeps.

Fast creeps typically move much faster than normal creeps. When slowed, they move the same as normal, non-slowed creeps. Slow-resistant creeps cannot be slowed. These two are similar in that they nullify strategies that rely on slowing, at least for those waves. In many games, a single wave can contain multiple types of creeps. Waves that contain both fast and regular speed creeps have natural group separation.

Flying creeps ignore walls and go straight to the exit. The paths are short and have no geometric properties (corners, U-Turns, etc.) to exploit. In most games with flying creeps, some towers cannot attack flying creeps, others cannot attack ground creeps and others can attack both, although potentially at different levels of damage.

Many other types of creeps exist, many with implications for strategies. Spawning type creeps create more creeps, with some having creeps that create child creeps (e.g., Ghost Hacker), others that split into multiple smaller creeps when attacked (e.g., Desktop Tower Defense). Some creeps have damage reduction and can only be hurt by strong weapons. Others have damage limits, taking only a few points of damage no matter how strong the tower. The former encourages the use of a few strong towers, the latter encourages the use of many small towers. Some games have creeps that can heal others. Some have invisible creeps that can only be seen if special detection towers are used.

In several games, creeps can change the path. Creeps that are killed might split in two, with one of the new creeps being thrown over a wall. In some, the walls of the maze are impassible terrain (water, canyons, etc.) and some creeps can create bridges over those. In one game, the Damage Over Time tower is an acid pool laid over the path. If the player uses more than one acid pool, a special acid creep will enter the first acid pool it encounters and come out the next one, potentially allowing the creep to move significantly ahead on the map while bypassing all the towers in between the two pools.

8.8.4 Targeting

In GopherTD, a tower selects the closest creep as its target and stays with it until the creep leaves its range. A later section discusses strategies for controlling which creeps are targeted.

In many games, the player has several options for explicitly controlling which creep is targeted. In Vector TD, the player can select for each tower whether the tower will attack the closest, strongest or weakest creep. Players can also turn sticky targeting on and off. In Desktop Tower Defense, different towers have different targeting methods, with some selecting the closest while others select those closest to the map exit. In Ghost Hacker, the player can manually select targets, instructing all towers in range to focus on the selected creep.

8.8.5 Mazing

There are two primary types of Tower Defense games, *mazing* and *non-mazing*. In non-mazing games, there are multiples maps, each with walls. Towers can only be placed on walls. In mazing games, there are no walls. Towers can be placed anywhere (although they are typically snapped to a grid position). Creeps follow the shortest path to the exit but cannot walk through towers, allowing the player to form mazes out of their towers. The strategies for forming mazes in mazing Tower Defense games have no corollary in non-mazing games such as GopherTD.

Chapter 9

Computer Model

One of the lessons drawn from the human studies is that problem solvers represent problem domains in terms of the strategies they afford. Given the right representation, identifying and executing strategies is fast and simple. Much of the work is done in the representation. We used these lessons to create computer agents capable of solving the same complex spatio-temporal tower defense problems that we saw humans solving. In this chapter we discuss our computational models of affordances, strategies and solver and how these were used to build a set of spatio-temporal reasoning agents.

9.1 Underlying Model

Figure 9.1 shows a class diagram of the AI system. The goal is to create a solution, which is a list of towers and the positions they should be placed on the map. In our implementation, the `Agent` does not create the solution. Instead, it determines which towers and strategies to use and stores this information in a `SolutionRequest`. This information is organized by functional groups. A functional group is a set of towers that share a goal. For example, in the DIFFERENTIAL SLOWING family of strategies (see

Section 8.2.3 To combat attack window density), the kill zone offense towers attempt to damage the creeps, the SLOW IN RANGE towers (see Section 8.2.1 To combat creep time in range) attempt to slow down the creeps while they are being attacked and the DIFFERENTIAL SLOWING towers attempt to separate the group of creeps by slowing one line of creeps and not the other. Each group has different placement strategies. Each group of towers and their set of strategies are stored in a `GroupStrategies` object. The solution request contains an ordered list of group strategies. The order in which groups are processed is important. Almost all humans placed offense towers first (except on "*fast*" maps, which are maps with short paths) and then placed slowing towers around them, which requires that the offense tower group is processed before the slowing tower group.

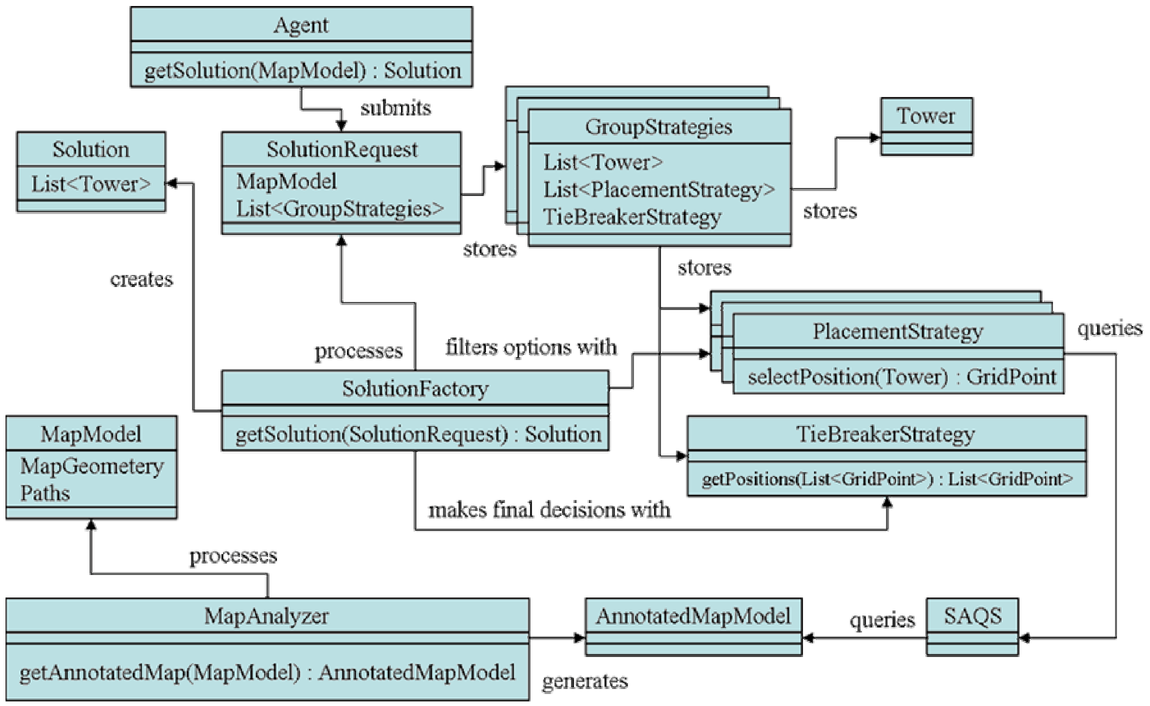


Figure 9.1. Class Diagram of the GopherTD AI. Represented as a Unified Modeling Language (UML) class diagram.

The `Agent` does relatively little work. The solution request is processed by the `SolutionFactory`, which is responsible for assembling a solution. It does not, however, perform the reasoning necessary to generate a solution. Instead, it processes each tower group by selecting a set of candidate positions (by default, all unoccupied wall tiles) and then, for each tower, asking each of that group's strategies to evaluate the candidate positions. The AI was implemented as a set of filters and roughly approximates the psychological theory of Elimination By Aspects (A. Tversky 1972). If a group has multiple strategies, the first strategy filters out any candidate position that does not meet the strategy's criteria. If more than one candidate remains, it is passed to the next strategy, continuing until there is only one candidate or all the strategies have been applied. If there is more than one candidate at the end, each group strategy set specifies a tie breaker strategy. The default tie breaking strategy simply selects a candidate at random. We assume that humans stop when they find the first acceptable position but we found it difficult to simulate this process computationally. By selecting a position at random, we assume that all options were equally likely to be selected. Our theory is that humans do not evaluate options in a purely random way but instead are guided by perception and preconception which, in turn, have rules. For example, it is possible that most humans begin by examining the center of a map and only move to the edges if nothing acceptable is found in the center. We did not have enough data on this process to form testable hypotheses and therefore chose not to model it in the computer agent.

The core of explicit reasoning is stored in the individual strategies. An implicit reasoning process is informally captured in the ordering and combination of strategies (a more explicit modeling of this form of intelligence is left as future work). Despite this, the reasoning at this step is normally quite limited. The majority of the strategies we observed were a single, simple rule such as "place towers where the path synchronization gap is highest". They also placed minimal demands on working memory.

No significant reasoning is done in the agent, solver or strategies. Where, then, is the intelligence? One option is that there is none – perhaps much of what is considered complex thought is nothing more than a set of simple decisions, with complex intelligence emerging from the interaction of a set of small intelligences. The explicit "run time" reasoning we observed in this project certainly appears to derive from the combination of a set of simple strategies. However, at least in this domain, these strategies would not be possible without the support of a rich, affordance-based representation of the problem. Consider the abstract strategy of divide and conquer – "Divide the problem into a set of small problems then solve the small problems". In this domain that strategy translates to "Separate the two lines of creeps and attack each line separately" (note that this is not a precise translation as the purpose of the DIFFERENTIAL SLOWING strategy in tower defense is to increase offensive power by reusing towers rather than reducing cognitive load, as is typically the case in divide and conquer algorithms). This strategy was not used by the many subjects (and none of the domain inexperienced subjects) because it was not obvious that it was relevant (we believe this is closely related to, but not identical to, the issue that it was not obvious how it could be applied). In contrast, all but one of the subjects who noticed (as captured in the think aloud protocol) that one line of creeps would sometimes fall behind the other (which we encode as the concept of path synchronization gap) immediately intuited the divide and conquer strategies EXPLOIT GEOMETRY and DIFFERENTIAL SLOWING. Determining that these strategies were relevant and how they could be applied depended predominantly on the subject's ability to recognize the path synchronization gap meta-feature, which in turn almost always led to the recognition of the actions this feature afforded.

9.1.1 Computational Model of Affordances

The `MapModel` contains the raw geometry and path data given to the agent by the system. The `MapAnalyzer` converts this to an `AnnotatedMapModel`. The `AnnotatedMapModel` contains meta-data about the map such as whether a tile is on a corner and the global step ID of a specific path tile (the global step ID says how many tiles the specified tile is away from the path start). It supports queries for lower-level data such as returning all wall tiles that do not have towers on them, are in range of a given path or past a certain spot on the map (as measured by the path, which represents the flow of time). When towers are placed, the tile meta-data includes information such as how many slowing towers have that tile in their range.

The `AnnotatedMapModel` is queried by the Spatial Affordance Query System (SAQS). `SAQS` allows the caller to request the existence, position and quality of spatial affordances and features on the map. Supported queries include `getPositionsByRelativeMaxValue` and `getPositionsByRelativePropertyValueAtLeast` which find or rank positions based on properties such as usable range, path synchronization gap and path coverage ratio (how much of a tower's range covers one path relative to the other path).

Decay queries allow the caller to see how a property decays over space or time. Consider the example where the best position for usable range is next to path tile 20 (i.e., the 20th from the start) and gives a range of 36. If one continues down the path, the best position after path tile 20 is at path tile 40, where the range is 32. The best position after that might be at position 60 with a usable range of 24. As you move further down the path, the value of the best positions drop (i.e., the best position between path tile 0 and 80 is higher than the best position between path tile 40 and 80). The query `getRegionTransitionPointsForMUR` returns a list of `ValueTransitionPoint`s for the MAXIMUM USABLE RANGE achievable by a specified tower, where a value transition point indicates where this value changes. This is useful for dividing the map into zones based on certain properties of interest to a strategy and assigning values to those zones, allowing the agent to evaluate tradeoffs for using a given zone for different purposes.

SAQS also supports functional queries such as `getPositionsForSlowingOneLineFullRange`, which returns the list of wall tiles where, if the specified tower were placed there, it could only attack creeps on one path. Additional queries include tasks such as geometric pattern matching (for agents who use rotation-invariant pattern-based strategies) and set operations such as determining which tiles are common to two different sets of tiles.

9.1.2 Computational Model of Strategies

When someone wants to know how to beat a game, ask for a raise or get a date, it is common for that person to ask someone successful "What's your strategy?". The presumption is that there is a single strategy, possibly complex and multi-step, to solve the problem. In this domain, such a strategy might look like the following:

- **Subject 3E0700**. Place all offense towers next to each other, preferably so that they form a square but making sure that every tower is next to the path so that it can use its full range (if you can't create a square this way, split the block so that it covers both sides of a path or flatten it into a line that follows the path). Place this block where it can cover the largest number of path squares. Surround it on all sides with slowing towers. This is your kill zone. Try to place it on the latter half of the map. This is so you can place a slowing zone in front of it that will separate the two lines of creeps, letting your kill zone deal with them one at a time.
- **Subject A0200**. Place the level 10 Red3 heavy offense towers where they cover the most path area. Because these towers have such a large range, this position is usually in the center of the map. Put all the towers next to each other so that their range circles cover almost identical areas. This will force them to choose the same target, allowing them to finish off individual targets faster. Try to separate the two lines of creeps by placing slowing towers where they can slow one line before it gets into range but won't slow the other one. Place slowing towers throughout the range of the heavy offense towers so that the creeps stay SLOW IN RANGE, giving the offense towers more time to attack. Place a few freezing towers near the end of the heavy offense towers' range to catch any creeps that manage to make it all of the way through. These towers will prevent the creeps from leaving the kill zone, giving the heavy offense towers a few more chances at killing them.
- **Subject N0100**. Use a pair of level 10 Red3 towers as these have the largest range. Place them where they can both cover a large amount of range and be passed multiple times. Place the towers in the same area, preferably right next to each other – the best positions tend to be next to each other and it's easier to share

resources. Once you place the heavy offense towers, place two level 10 Red1 towers in the same area. These towers aren't nearly as powerful but they're relatively cheap and they're fast. They'll support the heavy offense towers, wounding healthy creeps and finishing off the critically wounded. Since they have the same range as the heavy offense, place them so that they cover a large amount of range and will be passed multiple times. Once you place those, place a bunch of level 10 Green1 towers in the same area. You don't have enough to afford another Red1-10 tower along with all the slowing towers you want but you can get numerous Green1-10 towers for a fraction of the cost of one Red1-10. Since these towers have a small range, place them right next to the path. These towers are the least powerful towers in the game but they're so cheap you can afford half a dozen, so they're worth using to support the other towers. Once all these towers have been placed, place a bunch of slowing towers throughout the heavy offense towers' range to make sure the creeps stay slow while in attack range.

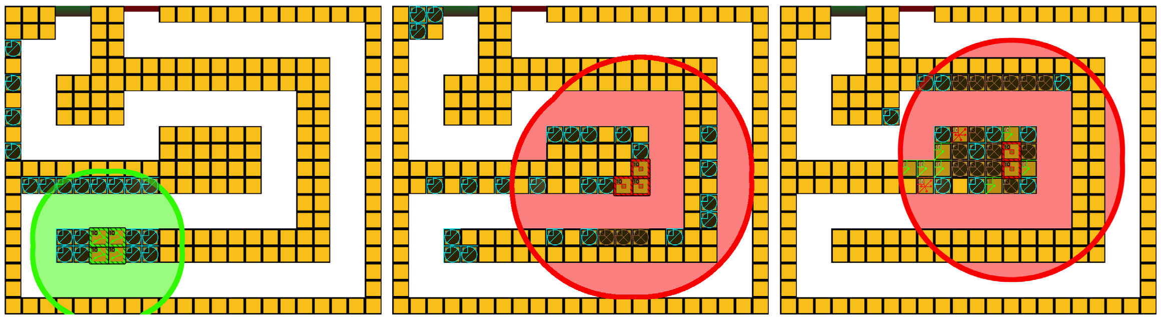


Figure 9.2. Offense Range on Elemental-ish. Solutions to the map Elemental-ish by subjects 3E0700 (left), A0200 (center) and N0100 (right). Combined range of heavy offensive towers shown.

Each of the above is comprehensive, complete and relatively complex. Each covers all aspects of solving a problem and can be used to create the full, final solution. Although many people use the term strategy to refer to these multi-step processes, we use the term *strategy* to refer to the individual, single-action steps inside them and use the term *play style* to refer to the collection of strategies. For example, for subject 3E0700, we define placing offense towers where they will cover the largest number of path squares as one strategy and surrounding those towers with slowing towers a second strategy. Assuming the computer model attempts to match strategies to problems, there are several reasons for using a fine-grained definition of strategy:

- **Applicability**. A single coarse-grained strategy is less likely to match/be fully compatible with a given problem. This is a serious issue as subjects rarely had a second strategy of this type (i.e., coarse-grained). By modeling a play style as a set of strategies, it is more likely that the subject will have at least some process that they can apply to the problem.
- **Conditionality**. The coarse-grained strategies above contain conditional, mutually exclusive pieces. For example, subject 3E0700 tries to place the heavy offense towers next to each other. If it is possible to arrange the towers in the form of a square while still having each tower be path-adjacent, do so, but if it is not, divide the towers into a pair of tower sets placed on either side of a path or, if that isn't preferable, convert it to a chain. Applying strategies of this sort is more complicated than applying a smaller, more focused strategy that recommends a single action. Finer-grained strategies do not necessarily decrease the amount of work involved but allow that work to be distributed across the matching and application processes.
- **Meta-Strategies**. The coarse-grained strategies, as described above, hide important details about how the pieces fit together. For example, what does it mean to "try to place the kill zone on the latter half of the map" (subject 3E0700)? The goal is to leave room for a DIFFERENTIAL SLOWING zone but this might prevent the kill zone from being placed at its best position. How much is the subject willing to lose from the kill zone to support DIFFERENTIAL SLOWING? Subject A0200 places some slowing towers throughout the heavy offense towers' range and others before the towers' range to create a DIFFERENTIAL SLOWING zone. Given a fixed number of slowing towers, how does the subject determine how many go to each area? These questions were not obvious to us during the human

studies, but when building the strategy instantiation capability of the computer agent it became obvious that we needed strategies for applying strategies.

- **Multiple, Independent Goals**. Although the strategies above are written in a way that implies the pursuit of a single goal, a detailed analysis of the human data showed that many strategies were created/learned in response to smaller, orthogonal goals. For example, the decision to place freezing towers at the end of a kill zone (A0200) was done to prevent leakage, a problem the subject noticed during execution failures. This strategy is specific to the problem of leaking and independent of A0200's play style. If it were adopted by the other subjects, it would easily integrate into their play styles and work equally as well (given an appropriate meta-strategy for resource allocation).
- **Incremental Learning**. We repeatedly observed subjects modifying their play style in both big and small ways in response to problems presented during the testing phase. When using fine-grained strategies, modeling learning is as easy as adding additional strategies to the subject's repository of known strategies and adjusting how strategies are chosen (in practice, every strategy we observed a subject learning was additive, meaning it did not replace a previous strategy and could therefore be selected without the need for a new strategy selection meta-strategy, although the subject did have to change how they allocated resources across strategies).
- **Transfer Learning**. Humans who have learned how to solve one problem can often solve new problems faster if the new problems meet one important criterion: the problems do not need to be similar in surface features or structure but they should offer the same affordances (i.e., a portion of the solution-generation process will be similar). Coarse-grained strategies are difficult to transfer to novel problems as many parts of them are unlikely to be relevant to the new problem. If, instead, the agent learns a set of fine-grained, single-purpose strategies, it is quite possible that a subset of the strategies will apply to a novel problem and help

solve the problem either on their own or in combination with strategies developed for other domains or created for the new problem.

- **Synthesis**. The field of artificial intelligence has several methods (case-based reasoning, policy adaptation, etc.) for adapting a solution to one problem to work on a new problem. This approach only works if the new problem and base problem are similar and the solution to the known problem is similar to the solution that must be generated. This approach requires a 1-to-1 mapping between known problems and new ones. In real life, humans often create solutions to new problems by drawing on lessons learned from a variety of problems. By modeling strategies as fine-grained and single-action, a set of strategies relevant to the current problem can be drawn from strategies learned from a variety of problems and combined in a novel way, making it easier to draw on experience to solve novel problems.

In this system we model coarse-grained play styles as a set of fine-grained, single-purpose, single-action strategies. The earlier strategies can be modeled as follows. Note that in addition to separating the rules we are also converting vague rules into more precise ones. We do this so that in a moment we can make an important observation point about the nature of these rules.

- **3E0700's play style**
    - Place the offense tower next to the other offense towers.
    - Place the offense towers where they achieve their MAXIMUM USABLE RANGE.
    - Place the SLOW IN RANGE tower next to an offense tower.
    - Place the kill zone after the DIFFERENTIAL SLOWING range.
    - Place the DIFFERENTIAL SLOWING tower so that it covers the path of the desired line of creeps and not the other.
    - Place the DIFFERENTIAL SLOWING tower before the kill zone.

    - Try to place it on the latter half of the map. This is so you can place a slowing zone in front of it that will separate the two lines of creeps, letting your kill zone deal with them one at a time.
- **A0200's play style**
    - Place the Heavy Offense tower where it achieves its MAXIMUM USABLE RANGE.
    - Place the range entrance of the Heavy Offense tower where it is closest to the range entrance of another Heavy Offense tower.
    - Place the range of the SLOW IN RANGE tower so that it covers a part of the Heavy Offense tower's usable range not already covered by another slowing tower.
    - Place the DIFFERENTIAL SLOWING tower so that it covers the path of the desired line of creeps and not the other.
    - Place the DIFFERENTIAL SLOWING tower before the kill zone.
    - Place the Cleanup tower's range so that it intersects the range exit of a Heavy Tower's last attack window.
- **N0100's play style**
    - Place the Heavy Offense tower where it achieves a normalized 50/50 balance of MAXIMUM USABLE RANGE and number of passes.
    - Place the Heavy Offense tower near another Heavy Offense tower.
    - Place the Heavy Offense tower in the same area as another Heavy Offense tower.
    - Place the Moderate Offense tower in the same area as a Heavy Offense tower.
    - Place the Moderate Offense tower near a Heavy Offense tower.
    - Place the Moderate Offense tower where it achieves a normalized 50/50 balance of MAXIMUM USABLE RANGE and number of passes.
    - Place the weaker (2nd group) Moderate Offense tower in the same area as a Heavy Offense tower.

- Place the weaker (2nd group) Moderate Offense tower near a Heavy Offense tower.
- Place the weaker (2nd group) Moderate Offense tower next to the path.
- Place the range of the SLOW IN RANGE tower so that it covers a part of the Heavy Offense tower's usable range not already covered by another slowing tower.

There are a few things to notice about the above strategy sets.

First, some of the strategies are unordered, others must be executed in a given order. For example, in order for N0100 to place a Moderate Offense tower next to a Heavy Offense tower, a Heavy Offense tower must be placed, meaning that the strategies for placing a Heavy Offense tower must be instantiated first. There are many ways to handle this. If strategies are represented as rules in a production rule system, the ordering, where necessary, is automatically handled by the system – rules that have another object as an antecedent are not activated/added to the agenda until the antecedent is fulfilled. This does not, however, handle situations where you want all towers in a given group to be placed first (e.g., the system could place one offense tower, then a slowing tower, then a second offense tower). For the work presented here, all towers were assigned to a group (`GroupStrategySet`), all groups had a sequence ID and all towers within a group were placed before any tower in the next group.

A second observation is that the strategies give potentially conflicting advice. For example, subject N0100 has three strategies for placing a heavy offense tower: 1) place them where they get the best compromise of usable range and number of passes, 2) place them near other heavy offense towers and 3) place them in the same area as other heavy offense towers. Note that the term *area* here means a contiguous and convex section of wall. The closest free position to a tower in one area is sometimes across the path in a different area, so stating that a tower must be near another tower does not guarantee that they are in the same area.

How do we handle strategies that give contradictory advice? This is less of a problem than it seems. The strategy instantiation system does not need to worry about this because the strategy selection system should filter out inherently contradictory strategies. For example, it would not allow the strategies "place tower near entrance" and "place tower near exit" because they are mutually exclusive. While the strategies listed above have the potential to be in conflict, in practice most do not (if they did, it is unlikely that the subject would have used them). This leads to the third observation – strategies act as filters. For example, consider the situation where a support tower must be placed using the strategies place the tower adjacent to a path, near a tower in the Heavy Offense group and in the same area as a tower in the Heavy Offense group. The first strategy would take in the set of wall tiles that do not already have towers on them and remove all positions that are not path adjacent. This filtered set of positions would then be passed to the next strategy which would return the set of candidate positions that were near (within a certain distance of) a tower in the specified group. This set of positions would then become the candidate set passed to the third strategy, which would return those candidate positions that were in the specified area. Note that, with the exception of strategies that maximize or minimize an ordinal property, the order of execution of these filters is irrelevant. For strategies where the order of execution is important, the order they are executed in is part of the agent's play style and is, in essence, a meta-strategy.

9.1.3 Solver

Human problem solving in this domain appeared to be an iterative, parallel process. Upon encountering a problem, subjects identified those actions that might help lead to a solution. These affordances were recognized by examining the problem solving strategies they knew, determining which affordances they required and examining the problem for those affordances. An examination of the tools available to them revealed actions which, paired with the problem environment and goals, suggested strategies to use. Once candidate strategies and the tools required to execute them were selected, the subject converted the abstract strategies into concrete actions, placing specific towers at specific locations in order to construct a solution.

The computer system we implemented acts a little differently. In our breadth experiments, subjects almost always used the same set of towers for each problem. To evaluate the impact of placement strategies, we required everyone in the performance experiment to use the same set of towers. For these reasons, there was no need to model the tool selection process. Figure 9.3 shows the implemented system

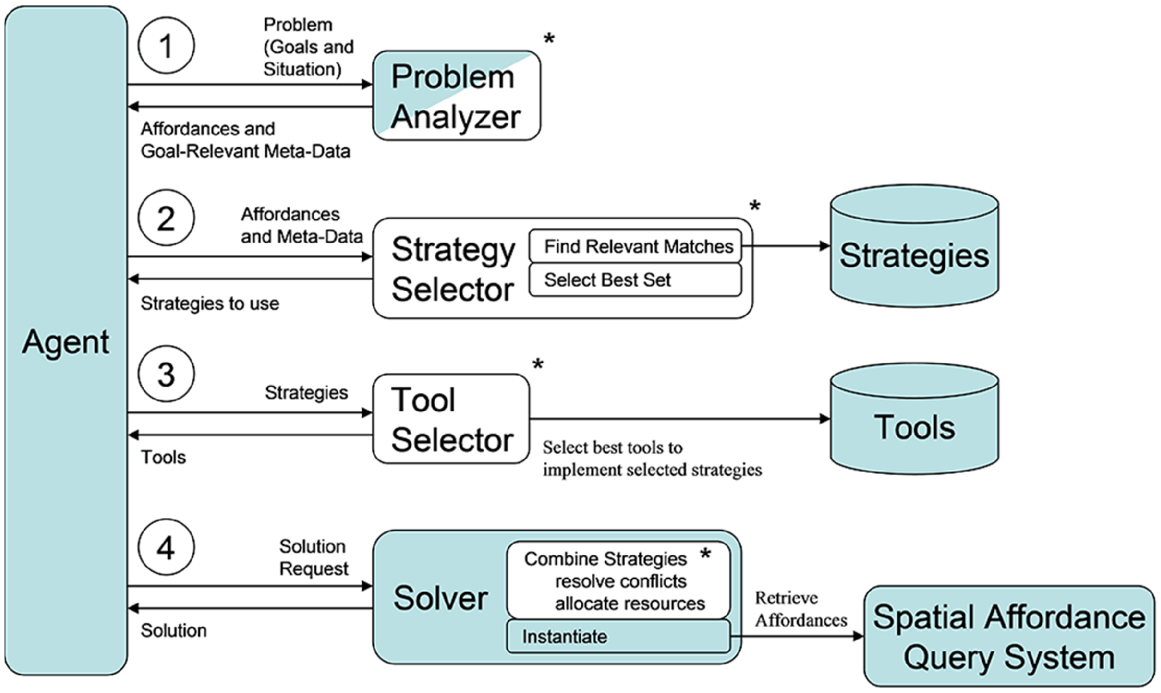

Agent
1
Problem (Goals and Situation)
Affordances and Goal-Relevant Meta-Data
Problem Analyzer
*
2
Affordances and Meta-Data
Strategies to use
Strategy Selector
Find Relevant Matches
Select Best Set
*
Strategies
3
Strategies
Tools
Tool Selector
*
Select best tools to implement selected strategies
Tools
4
Solution Request
Solution
Solver
Combine Strategies resolve conflicts allocate resources
*
Instantiate
Retrieve Affordances
Spatial Affordance Query System

Figure 9.3. Implemented System. Our implementation of the previously mentioned conceptual model. The two computational goals in this project were to emulate specific people and measure the quality of different strategies. Because the processes in the white boxes were not relevant to those goals, those portions of the system were not implemented.

Because the towers and placement strategies were fixed, there was no need to use an iterative process to solve the problem. Humans only identified meta-data and affordances relevant to the strategies. The computer model does not. Humans presumably search for less information because problem analysis of this type is expensive. Identifying all meta-data that might be used by any known strategy, regardless of whether the strategy was known by the agent, is a subsecond operation for the computer agent. We saw no other reason not to gather this data so the GopherTD problem analyzer does not satisfice in this process.

To understand the solution process, it is important to understand a few key ideas.

Section 7.3 Placement Decision discusses the idea of *placement decisions*. A placement decision says a Placement Target will be placed with a given Placement Relationship to one or more Placement Anchors. For example, the strategy "place towers along the path" can be represented by the placement decision [Target: Tower, Relationship: Touching, Anchor: Path]. Strategies are kept general by using meta-data rather than concrete positions. This meta-data is resolved by the Spatial Affordance Query System at run time.

Continuing the example from the previous section, we can represent N0100's strategies as:

- **Functional Group: Heavy Offense (group ID: 1)**
    - Place tower where it achieves a normalized 50/50 balance of MAXIMUM USABLE RANGE and number of passes.
        - Target: Tower
        - Relationship: On
        - Anchor: Combination(Usable Range, Number of Passes)
    - Place tower in the same area as another Heavy Offense tower.
        - Placement=[Target: Tower, Relationship: On, Anchor: Area (Group 1)]
    - Place tower near another Heavy Offense tower.

        - Placement=[Tower, Near, Set of Towers (Group 1)]
- **Functional Group: Moderate Offense (group ID: 2)**
    - Place tower where it achieves a normalized 50/50 balance of MAXIMUM USABLE RANGE and number of passes.
        - Placement=[Tower, On, Combination(Usable Range, Number of Passes)]
    - Place tower in the same area as Heavy Offense towers.
        - Placement=[Tower, On, Area (Group 1)]
    - Place tower near Heavy Offense towers.
        - Placement=[Tower, Near, Tower Group (Group 1)]
- **Functional Group: Light Offense (group ID: 3)**
    - Place tower in the same area as Heavy Offense towers.
        - Placement=[Tower, On, Area (Group 1)]
    - Place tower near Heavy Offense towers.
        - Placement=[Tower, Near, Tower Group (Group 1)]
    - Place tower next to the path.
        - Placement=[Tower, Touching, Path]
- **Functional Group: Slow in Range (group ID: 4)**
    - Place the range of the tower so that it covers part of a Heavy Offense tower's usable range.
        - Placement=[Range, Intersect, Range (Group 1)]
    - Place the range of the tower so that it covers path area not already covered by another slowing tower.
        - Placement=[Range, On, Uncovered Path]

In this domain, strategies are filters. They take a set of candidate options and return those that match the specified criteria. In our definition of strategies, a strategy can only represent a single placement decision. N0100's approach to using low-powered supporting offense towers is to place them along the path in the same area as the heavy offense towers. This requires two placement decisions: [Tower, Touching, Path] and [Tower, In, Area(Heavy Offense towers)]. We consider these to be two separate strategies. Each strategy is a filter and these filters can be chained. Implementing strategies as a set of filters allows us to model complex strategies that are a union or prioritization of different criteria while leaving us with the flexibility to achieve several goals such as run-time strategy creation and integration with other strategy sets. We presume these capabilities are necessary to achieve transfer learning in domains that require new strategies but benefit from integrating existing ones. It is also theorized that modeling strategy sets as placement decisions, which have a fixed format and verifiable contents, is critical to the advanced user modeling problem of strategy inference.

Strategies specify how to place a single tower. They do not operate on sets of towers. In order to place a set of towers, the strategies must be applied to each tower. Different towers use different strategy sets. To understand how this mapping is maintained, it is important to realize that towers belong to a hierarchy of groups. An example is shown in Figure 9.4. The subject decides whether they will, at a high level, divide the map into pieces. Some subjects placed all their towers in a single kill *zone*, others divided them between multiple kill zones or multiple areas with different goals (this was most commonly seen in the DIFFERENTIAL SLOWING family of strategies). Within these zones, towers were grouped into *roles* (e.g., primary offense, supporting offense, slowing, cleanup, etc.). Within these roles, towers were grouped into functional groups. A *functional group* is a group of towers with the same goals that will be placed using the same strategies. Strategies are therefore stored at the functional group level.

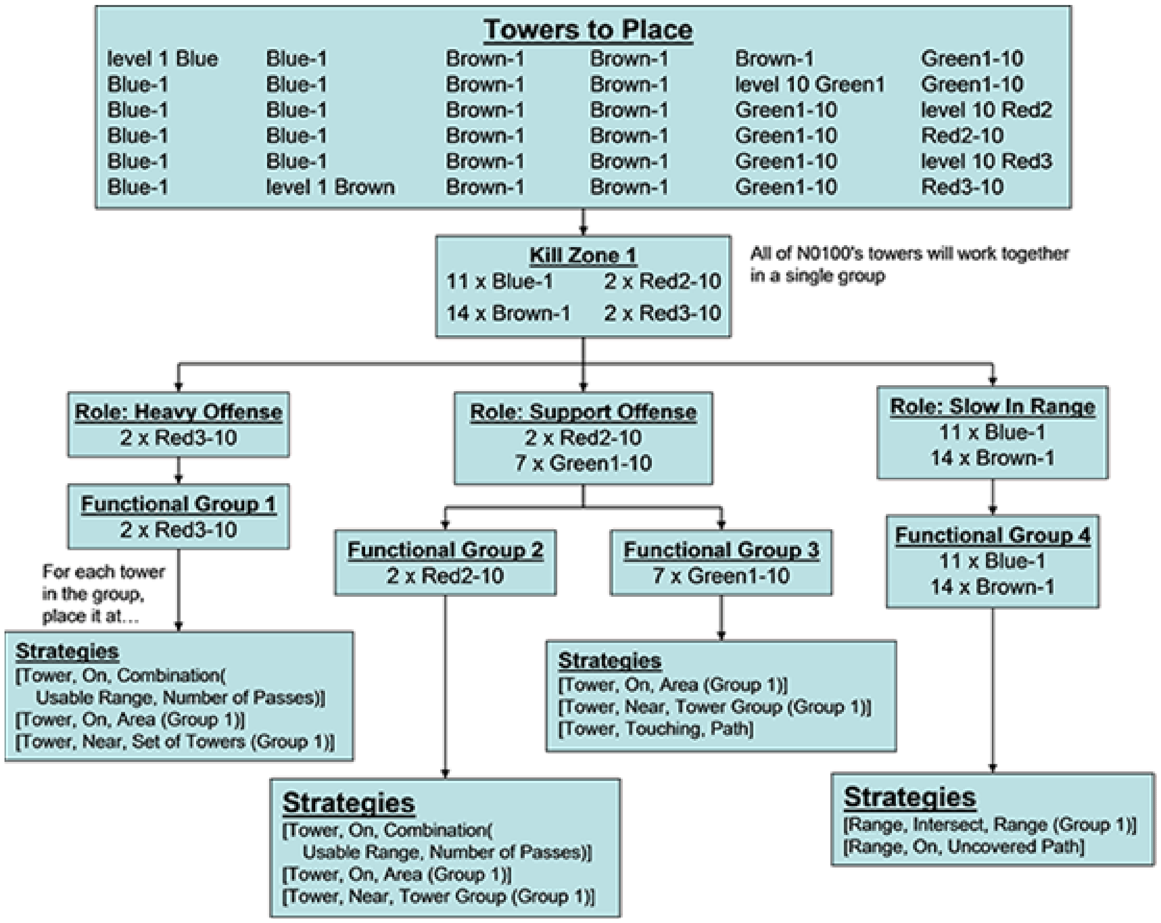


Figure 9.4. N0100's Towers. The solution process followed by subject N0100. The subject's task is to place 29 towers on the map (the process for selecting towers is not shown here). The subject decides to place all their towers in a single area (the kill zone). They divide towers into three roles: Heavy Offense, Moderate Offense and SLOW IN RANGE. The Moderate Offense towers are further divided into two functional groups; each fulfills the role of Moderate Offense but they are placed in different ways. Each functional group has an ID that represents its priority; all towers in group 1 are placed, then the towers in group 2, etc. Each tower in the functional group is placed using the placement strategies for that group. These strategies are placement decisions that operate on meta-data. N0100's strategies for low powered Moderate Offense towers (group 3) say to place them along the path in places near the Heavy Offense towers.

Because of the goals of this work, GopherTD problem solving agents assume they are given a set of towers to place and will use a fixed set of strategies. Towers are assumed to be organized into functional groups and the functional group IDs accurately reflect any dependency relationships (i.e., towers in group 2 may ask to be placed near towers in group 1 since group 1 is placed first but towers in group 1 may not reference group 2).

To generate a solution, the agent selects the first functional group and the strategies associated with it. It then selects a tower from that group and applies the first placement strategy. This returns a set of candidate positions. This set is passed to the next placement strategy, continuing until all of the placement strategies have been executed. Since multiple positions might match the strategy's criteria, a tie breaker strategy is used to pick a single position. This process is repeated for the next tower in the group. Once all towers in the functional group are processed, the next functional group is selected and the process repeats. The class diagram for the relevant classes is show in Figure 9.5.

```
public getSolution(SolutionRequest request) returns Solution
{
   Solution solution = new Solution()
   AnnotatedMapModel map = MapAnalyzer.getMapAnalysis(request.map)
   for (int groupID = 0; groupID < request.groupStrategies.Count
        groupID++)
   {
      GroupStrategySet group = request.groupStrategies[groupID]
      foreach (TowerLog tower in functionalGroup.towers)
      {
         List<GridPoint> candidatePositions =
            map.getPositionOfFreeWallCells()
         foreach (PlacementStrategy strategy in
               group.placementStrategies)
         {
            candidatePositions =
               strategy.getPositions(map, candidatePositions)
         }
         GridPoint position =
            group.tieBreaker.getPosition(candidatePositions)
         solution.addTower(tower, position, groupID)
      }
   }
   return solution
}
```

Listing 9.1. Solver Code. Given a `SolutionRequest`, the `Solver` loops through each functional group and, for each tower in the group, applies the selected strategies. Rather than solve problems directly, the solver delegates placement decisions to the strategies.

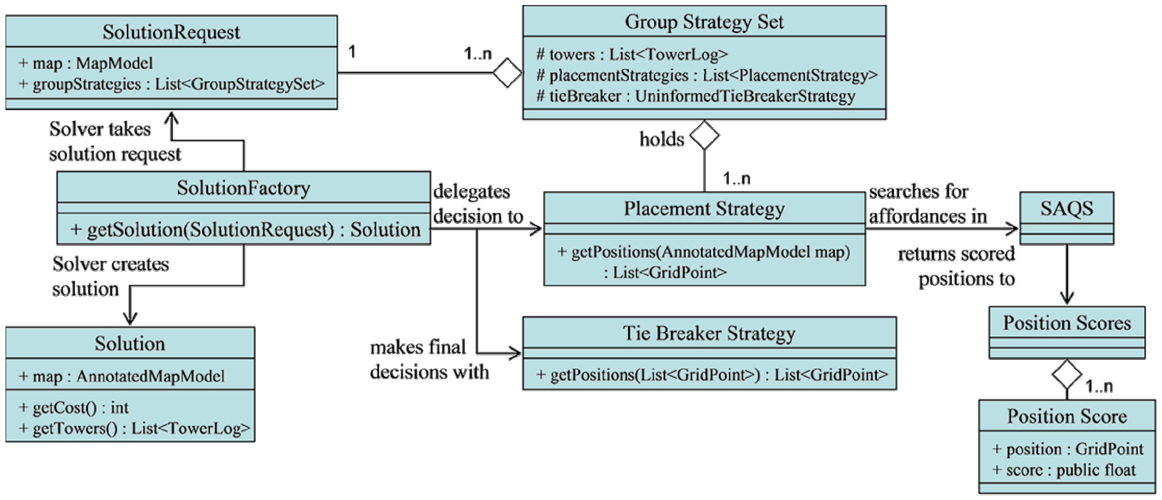

SolutionRequest
+ map : MapModel
+ groupStrategies : List<GroupStrategySet>
1
1..n
Group Strategy Set
# towers : List<TowerLog>
# placementStrategies : List<PlacementStrategy>
# tieBreaker : UninformedTieBreakerStrategy
Solver takes solution request
holds
1..n
SolutionFactory
+ getSolution(SolutionRequest) : Solution
delegates decision to
Placement Strategy
+ getPositions(AnnotatedMapModel map) : List<GridPoint>
searches for affordances in
SAQS
returns scored positions to
Solver creates solution
Solution
+ map : AnnotatedMapModel
+ getCost() : int
+ getTowers() : List<TowerLog>
makes final decisions with
Tie Breaker Strategy
+ getPositions(List<GridPoint>) : List<GridPoint>
Position Scores
1..n
Position Score
+ position : GridPoint
+ score : public float

Figure 9.5. Class Diagram for Strategy System.

The specific technical process is as follows:

1. The agent creates a request for a solution, `SolutionRequest`. This request includes the problem (in this domain, the map geometry) and a set of `GroupStrategySet` objects. There is one `GroupStrategySet` object for each tower functional group and it explains how the towers in that group should be placed. Each `GroupStrategySet` contains the following:
   a. **Tower Functional Group**. The towers in this functional group.
   b. **Placement Strategies**. An ordered set of strategies to use to place each tower in the functional group. Each strategy takes an unordered set of candidate positions and returns a filtered version of that set.
   c. **Tie Breaker Strategy**. Selects a specific position from a set of candidates.
2. The `SolutionRequest` is passed to the solver, `SolutionFactory`.
3. The `SolutionFactory` creates a solution for the given problem by assigning each tower to a position in the `Solution`. It does this by iterating through the functional groups, towers in those groups and strategies for those towers

```
for each functional group (i.e., GroupStrategySet)
   for each tower in the functional group
      candidate positions = the list of unoccupied wall tiles
      for each placement strategy in the functional group
         input: set of candidate positions
         filter out candidates that do not meet the
```

```
                strategy's criteria
            candidate positions = newly filtered set of
                candidates
        if there's more than one candidate left, use a
            tie breaker
        add to the solution the specified tower at the
            calculated position
```

4. The position to place each tower is stored in a `Solution` object, which is passed back to the agent.

The code for this solution creation is given in Listing 9.1. It is worth noting that very little of the agent's intelligence is in the solver. Rather than solve problems directly, the solver delegates placement decisions to the strategies. The candidate set of locations for the tower begins as the set of unoccupied wall cells and then each placement strategy narrows this set down. No single strategy makes the placement decision and thus the intelligence of the system does not exist in any one place, it is distributed throughout the system.

9.1.3.a Strategy Implementation

A strategy specifies an action over abstract data. In this domain, a placement strategy represents a placement decision where each part of the decision (target, relationship and anchor) is an abstraction, feature, affordance or meta-data. Consider the strategy "place the tower on a corner". If an agent wishes to use this strategy, they indicate this with the following code:

```
PlacementStrategy strategy =
   new PlaceTowerOnAbsolutePropertyStrategy(
      MapPropertyAbsolute.Corner,
      StrategyOperator.Maximum);
placementStrategies.Add(strategy);
```

The strategy "place the tower so that it covers as much of the path as it can" is written as:

```
PlacementStrategy strategy =
   new PlaceTowerOnRelativePropertyStrategy(
      MapPropertyOrdinal.PathCellsInRange,
      StrategyOperator.Maximum);
placementStrategies.Add(strategy);
```

The agent maintains a collection of strategies to use for towers in different functional groups. During problem solving, the strategy is converted to a concrete action. The first part of the instantiation process is for the agent to specify the data the strategy will work with. For the MAXIMUM USABLE RANGE strategy just mentioned, the strategy needs to know the map, the set of candidate positions on the map and the tower to place on the map. The code to invoke this process looks like:

```
candidates = strategy.getPositions(tower, map, candidate);
```

As mentioned in the previous section, `candidates` is the set of positions the strategy will consider. It is initially set to all wall tiles on the map that do not already have a tower on them. Each strategy takes this set and removes those candidates that do not match the specified criteria. In the following code:

```
List<GridPoint> candidatePositions =
   map.getPositionOfFreeWallCells();
candidates = cornerStrategy.getPositions(map, candidate);
candidates = usableRangeStrategy.getPositions(
               tower, map, candidate);
```

The candidate set begins as all free wall tiles. The first strategy removes all positions that are not on a corner. The second strategy takes this reduced set and returns the corner positions where the tower will cover the most path area. The result is a set of positions that are on unoccupied corner wall tiles where the tower will cover the most path area.

While there are thousands of possible placement strategies, specifying them only requires a few different interfaces. These interfaces differ in what data the agent must specify. The different interfaces used in GopherTD are shown in Figure 9.6.

All placement strategies need to know about the map. This is sufficient for finding absolute properties, which are properties whose definition does not depend on context. Examples include spatial features such as corners and exterior walls (the walls surrounding the border of the map) and meta-properties / affordances such as the positions where multiple paths split, join and overlap. These strategies were specified using the `PlaceTowerOnAbsolutePropertyStrategy` class.

Most of the strategies we identified relied upon properties whose definitions depend on the tower being placed. Examples include U-turns, islands, usable range, path synchronization gap and number of passes. These strategies were specified using the `PlaceTowerOnRelativePropertyStrategy` class.

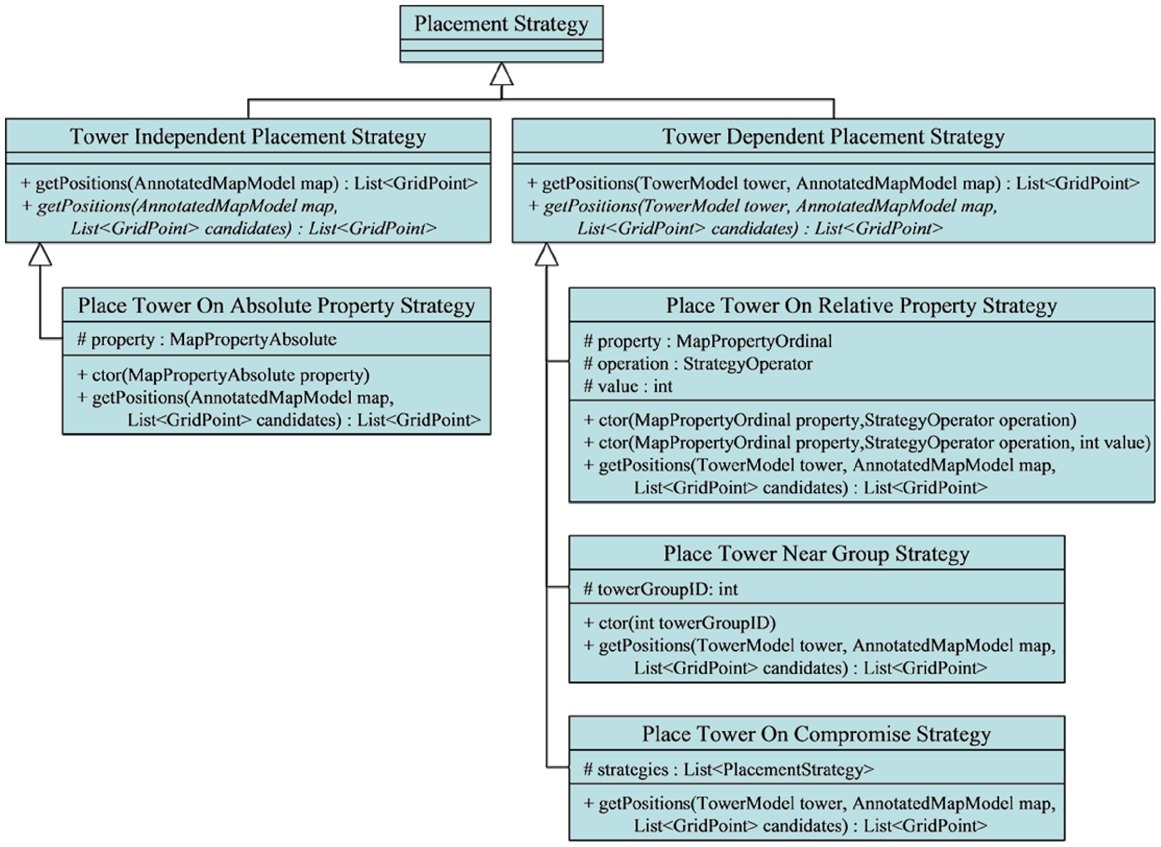

Placement Strategy
Tower Independent Placement Strategy
+ getPositions(AnnotatedMapModel map) : List<GridPoint>
+ getPositions(AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>
Tower Dependent Placement Strategy
+ getPositions(TowerModel tower, AnnotatedMapModel map) : List<GridPoint>
+ getPositions(TowerModel tower, AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>
Place Tower On Absolute Property Strategy
# property : MapPropertyAbsolute
+ ctor(MapPropertyAbsolute property)
+ getPositions(AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>
Place Tower On Relative Property Strategy
# property : MapPropertyOrdinal
# operation : StrategyOperator
# value : int
+ ctor(MapPropertyOrdinal property,StrategyOperator operation)
+ ctor(MapPropertyOrdinal property,StrategyOperator operation, int value)
+ getPositions(TowerModel tower, AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>
Place Tower Near Group Strategy
# towerGroupID: int
+ ctor(int towerGroupID)
+ getPositions(TowerModel tower, AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>
Place Tower On Compromise Strategy
# strategies : List<PlacementStrategy>
+ getPositions(TowerModel tower, AnnotatedMapModel map, List<GridPoint> candidates) : List<GridPoint>

Figure 9.6. Class Diagram for Placement Strategies. Strategies vary in the information they require from the user.

The majority of strategies were instances of `PlaceTowerOnRelativePropertyStrategy`. `PlaceTowerNearGroup` was used for strategies such as "place the tower near the heavy offense towers" where the agent had to specify an additional tower or group of towers. `PlaceTowerOnCompromiseStrategy` was used for meta-strategies that combined the results of two or more strategies run in parallel (e.g., "place the tower at a position that is a good balance of useable range and path synchronization gap").

Placement strategies always return a set of positions. To select the actual position to use, a tie breaker strategy is used. Most agents in GopherTD broke ties by selecting a random position.

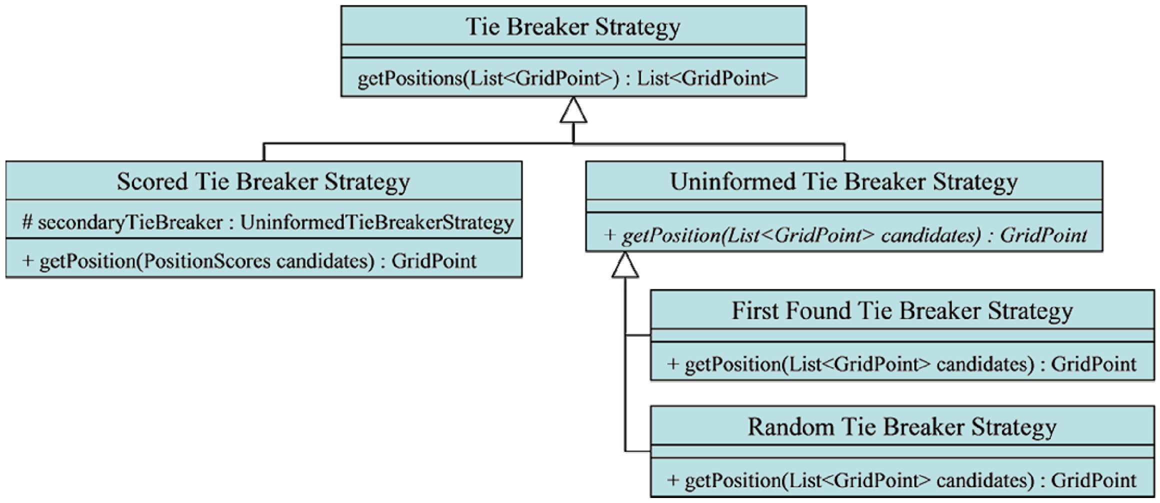

Tie Breaker Strategy
getPositions(List<GridPoint>) : List<GridPoint>
Scored Tie Breaker Strategy
# secondaryTieBreaker : UninformedTieBreakerStrategy
+ getPosition(PositionScores candidates) : GridPoint
Uninformed Tie Breaker Strategy
+ getPosition(List<GridPoint> candidates) : GridPoint
First Found Tie Breaker Strategy
+ getPosition(List<GridPoint> candidates) : GridPoint
Random Tie Breaker Strategy
+ getPosition(List<GridPoint> candidates) : GridPoint

Figure 9.7. Class Diagram for Tie-Breaking Strategies. Placement strategies return a set of options. Tie breaking strategies return a single option from that set.

Placement decisions are of the form "place the placement target in the specified placement relationship to a placement target". Examples range from the simple:

- Place the tower on a corner
- Place the tower on a U-turn

to the more complicated

- Place the tower's range's entrance before the range exit of a heavy offense tower's last attack window
- Place the tower's first attack window where it covers part of path 1 but none of path 2

- Place the tower's range so that it covers a path area that is already covered by a slowing tower
- Place the tower's range's exit where at least some of the creeps will be moving slowly

Strategies use abstractions but do not define them. It is the job of the Spatial Affordance Query System to decide which positions on the map match the requested criterion. There is very little code needed for an agent to specify its sets of strategies, a strategy to specify its actions or the solver to instantiate and apply a strategy. Most of the AI code resides in the systems that define the affordances and meta-data (i.e., the classes AnnotatedMapModel, SAQS and the classes used by SAQS).

In humans we observed that the portions of problem solving that relied on memory (strategies) and processing power (reasoning) were, to use the terminology of Gigerenzer (Gigerenzer and Goldstein 1996; Bertel and Kirlik 2010), fast and frugal. Strategies were small, focused and representationally simple and reasoning was straight forward and computationally simple. That a broad class of complex problems can be solved using a lightweight memory and processing model is possible because the majority of the system's complexity is hidden in how the environment is represented. A useful representational model for solving spatio-temporal problems appears to be innately guided andexperientially acquired by most healthy adult humans, or at least was present in all of the subjects we studied, including those who had no experience with our domain or any domain similar to it. We also observed that experience allowed subjects to learn better models and that better models led to better strategies and consequently better performance.

Our computer model exhibits similar properties. The strategies are simple to specify and easy to apply so long as there is a strong representational system to support them. Sets of strategies and a generic representational system also make it possible to leverage existing knowledge to solve problems that are not merely novel but are significantly, structurally dissimilar, a class of problems not currently solvable by existing AI techniques. By pushing the majority of the "intelligence" into the representational system and using light-weight strategies keyed to those representations, the GopherTD system is able to solve novel, complex problems in sub-second time. While we feel this points to a very successful computer model of problem solving in a domain computers traditionally do poorly in, it would not have been possible without knowing which abstractions to model and that could not have been done without the human studies. We do not know of a way that the computer could have learned these abstractions on its own, although it should be noted that it is not clear that these abstractions are learnable; it is possible and perhaps probable that they are, at least at some base, boot-strap level, innate in humans and a product of evolution rather than something entirely learned from experience.

9.2 Baseline Agents

In this section we discuss three baseline agents.

The Random Agent places towers at random positions. It has no heuristics and uses no meta-data or abstract concepts.

The Path Adjacency Agent places towers next to path tiles. It uses a simple filtering heuristic to remove all tiles that do not match the stated criteria (e.g., not an unoccupied wall tile next to a path tile). This heuristic is based on the idea that locations near the target (i.e., the creeps) are better than locations far away, which is a general piece of spatial knowledge.

The MAXIMUM USABLE RANGE Agent places towers where they cover the largest number of path tiles. It uses a straight-forward scoring strategy based on the property usable range and the hypothesis that more space means more time for the tower to attack which means higher scores. Usable range is a property not directly represented in the raw data but which can be concretely measured using the map geometry and path data. The space=time hypothesis is a higher-level, abstract, spatio-temporal hypothesis. In an informal survey, this was the strategy suggested by the majority of people who were shown GopherTD, had no prior experience with the tower defense domain and did not participate in the study and thus had no opportunity to learn from experience. This strategy seemed obvious to this group of people and was normally stated within a minute of seeing the domain. Despite this, there is no obvious way for the computer to come up with this strategy on its own. The development of this strategy requires general spatio-temporal knowledge that humans appear to have independent of domain experience and computers do not.

No human subject solved problems in the manner of these three agents. While those not participating in the experiments generally suggested the MAXIMUM USABLE RANGE strategy, experiment subjects, who used training time to test their strategies, used a more sophisticated set of strategies, which we collectively label the ADVANCED MAXIMUM USABLE RANGE approach. An agent implementing this is described in Section 9.3.1 Advanced Maximum Usable Range – The Obvious Strategy.

9.2.1 Random – Measuring Chance Performance

As expected, the Random Agent places towers at random positions on the map. The positions must be walls and cannot already be occupied by another tower. The agent can be made to select towers and upgrade levels at random, select random towers based on criteria (i.e., only red3 towers from level 4-7 and level 3 brown towers) or use a specified set of towers.

The Random Agent acts on raw data. It has no concept of meta-properties such as the number of path cells in range, the amount of time the tower will be active, functional roles, grouping, etc. An important implication is that randomly placed towers can be placed at positions where they cover no path cells and therefore never activate. For this reason, the random agent's performance is strongly correlated to the range of the towers used and the percentage of wall tiles that are inside that tower's range. When using short range towers such as the Green3-10 and Blue-1 towers used in phase 3, random does better when a significant percentage of tiles are near a path. Examples are shown in Figure 9.8 and Figure 9.9.

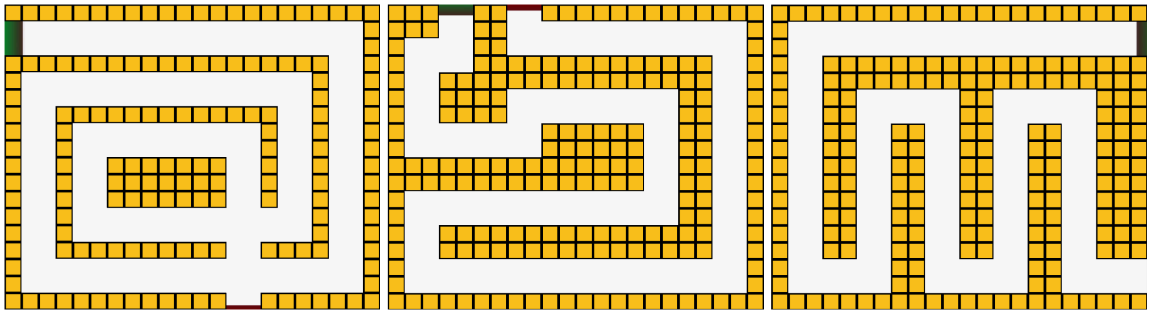


Figure 9.8. Best Maps for Random. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the Random Agent scores highest on these three maps. From left to right: 27/28, 26/28, 25/28.

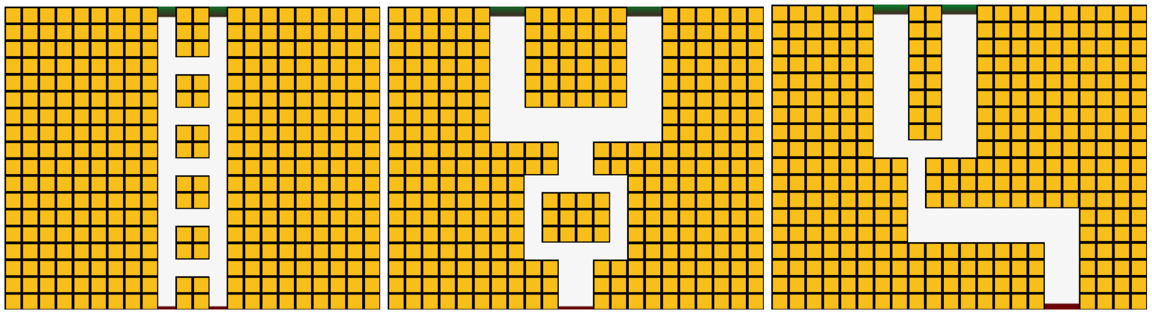


Figure 9.9. Worst Maps for Random. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the Random Agent scores lowest on these three maps. From left to right: 3/28, 3/28, 3/28.

It is worth noting that the hardest and easiest maps for the Random Agent are the same as those for humans. The human scores are significantly higher (see D.3.1Effectiveness of Spatial-Proxying) but the maps with a lower percentage of path adjacent tiles are still the most challenging ones. At first this might seem odd – humans should be smart enough to avoid those tiles where the tower will have no range. The answer likely lies elsewhere. In this domain, where the map is quite small (22x18), for more tiles to be near a path the path must be longer and for a longer path to fit in the available space it must loop back on itself, presenting an opportunity for well placed towers to have multiple attack windows. Maps with fewer path-adjacent tiles typically have shorter paths and fewer loop-backs, removing opportunities for strategic players.

9.2.2 Path Adjacency – A Simple Heuristic

The Path Adjacency Agent uses a path adjacency filter strategy to reduce the candidate set to those tiles directly adjacent to a path tile. Once this candidate set is established, the agent behaves the same as the random agent.

The path adjacency filter rules out any locations where a tower would have no usable range. Since being next to a path tile results in a larger usable range than a position one tile back from the path, this strategy behaves somewhat like a MAXIMUM USABLE RANGE strategy. It trades guarantees for simplicity – placing a tower at a random tile next to the path does not guarantee that the tower will obtain the MAXIMUM USABLE RANGE but it does not use any memory or rely on any thought process beyond perception, unlike a MAXIMUM USABLE RANGE strategy which must measure usable range, compare positions and remember the best matches.

The Path Adjacency Agent selected all unoccupied wall tiles next to a path tile. This strategy does not require the agent to know anything about the tower to be placed. The strategy used is a subtype of the class `PlaceTowerOnAbsolutePropertyStrategy`, where "absolute property" means that the value of the property (in this case, adjacency to path tiles) is not dependent on context.

```
PathAdjacencyAgent::getSolution(MapModel map) returns Solution
{
   SolutionRequest.MapModel = map
   SolutionRequest.towers   = getTowersToPlace()

   SolutionRequest.groupStrategies.placementStrategies.Add(
      new PlaceTowerOnAbsolutePropertyStrategy(
         MapPropertyAbsolute.PathAdjacent))

   SolutionRequest.groupStrategies.tieBreaker =
      RandomTieBreakerStrategy
   return SolutionFactory.getSolution(SolutionRequest)
}
```

Listing 9.2. Path Adjacency Agent.

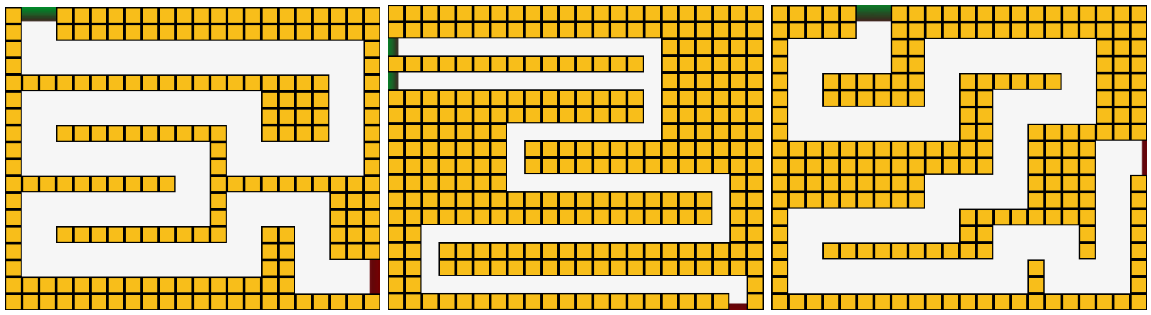


Figure 9.10. Best Maps for Path Adjacency. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the Path Adjacent Agent scores highest on five maps, 26/28. This includes all three maps shown in Random Agent and the ones shown here (right most map earned a score of 25/28).

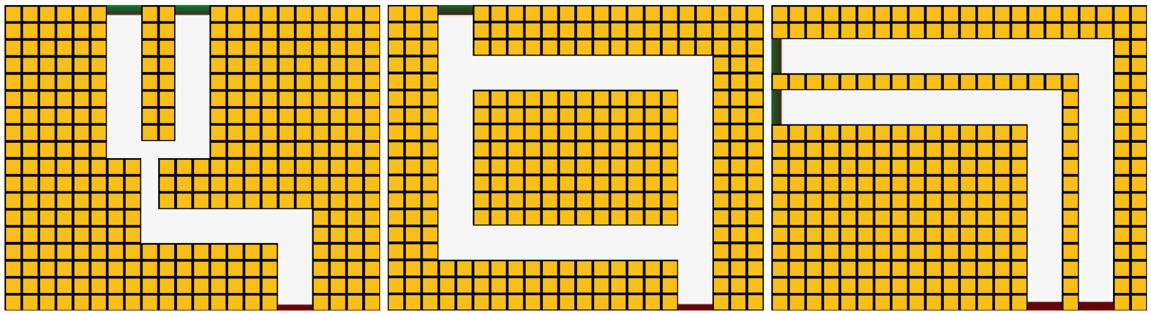


Figure 9.11. Worst Maps for Path Adjacency. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the Path Adjacent Agent scores lowest on these three maps. From left to right: 17/28, 13/28, 13/28.

9.2.3 Maximum Usable Range – A Purely Spatial Strategy

The Path Adjacency Agent used a single filter strategy and selected at random from those that passed the filter. The MAXIMUM USABLE RANGE Agent uses a single scoring strategy, rating each unoccupied wall tile by the amount of path space that can be covered by a tower placed there.

There are many ways to measure usable range. This agent uses virtual path tiles, which looks beyond physical path tiles to path tile traffic. If a path tile is traversed by more than one line of creeps or more than once by a given line of creeps, the reported number of path tiles is increased. Although this does not match the number of physical, visible path tiles, it is a better indicator of the number of creeps that pass through a tower's range and the amount of time they spend there. Not all human subjects used this more functional version of path area. A discussion of alternative methods of measurements is given in Chapter 8 Strategies.

```
MaximumUsableRangeAgent::getSolution(MapModel map) returns Solution
{
   SolutionRequest.MapModel = map
   SolutionRequest.towers   = getTowersToPlace()

   SolutionRequest.groupStrategies.placementStrategies.Add(
      new PlaceTowerOnRelativePropertyStrategy(
         MapPropertyOrdinal.PhysicalPathCellsInRange,
         StrategyOperator.Maximum))

   SolutionRequest.groupStrategies.tieBreaker =
      RandomTieBreakerStrategy
   return SolutionFactory.getSolution(SolutionRequest)
}
```

Listing 9.3. Maximum Usable Range Agent.

The MAXIMUM USABLE RANGE Agent places a tower at the position where it will cover the most path area. How much path area is in range depends on the range of the tower being placed. This is not merely a matter of scaling – the top 10 positions for one tower are not guaranteed to be the top 10 positions for a tower with a different sized range.

Usable range property is not an absolute property, its value depends on context, in this case the tower to be placed. The strategy used is of type `PlaceTowerOnRelativePropertyStrategy`, where "relative property" means the value of the property is relative to or dependent on other information.

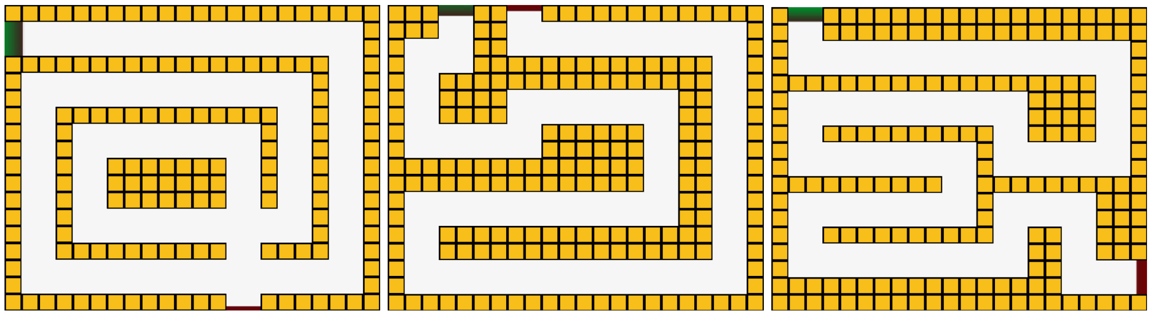


Figure 9.12. Best Maps for MAXIMUM USABLE RANGE. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the MAXIMUM USABLE RANGE Agent averages a perfect score on one map (left) and 27/28 on five other maps. The center and right images are two of those five.

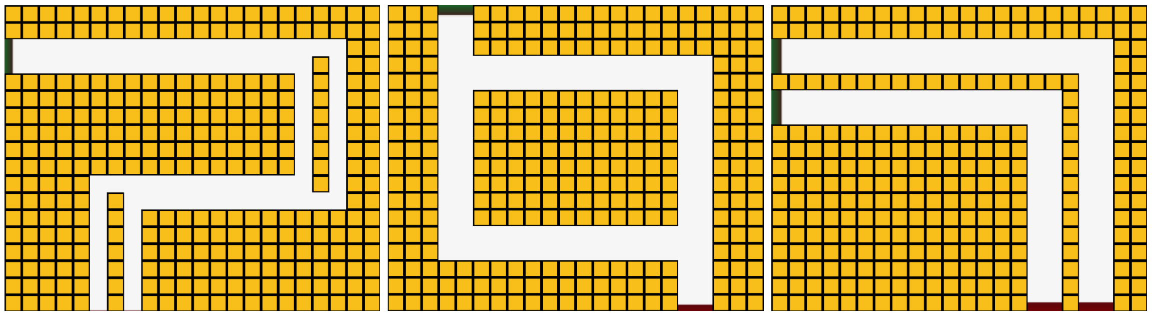


Figure 9.13. Worst Maps for Maximum Usable Range. When using the Experiment 2 towers (four Green3-10, 20 Blue1-1), the MAXIMUM USABLE RANGE Agent scores lowest on these three maps. From left to right: 17/28, 14/28, 13/28.

9.3 Strategy Agents

Through the course of this project we created a variety of computer agents. *Simulation agents* were created for several human subjects. Each of these agents replicated the problem solving style of a specified person. The goal of these agents was to accurately simulate the behavior of a given person, including errors and shortcomings. While we consider this work to be important for many reasons, including helping us gather, understand and validate our data on problem solving, representations and strategies, the focus of this document is on efficient, optimal agents, agents that can match the best humans in terms of solution quality, processing time and generalization.

This section focuses on *strategy agents*, agents that replicate a specific strategy or set of related strategies. While these strategies were derived from human data and generate solutions similar to what some humans might generate, the computer models made no attempt to replicate human processes. They are, in a sense, behaviorally accurate but potentially cognitively inaccurate.

The strategies we modeled were those that were used by a significant percent of the subjects and had a significant impact on scores. This includes what might be considered primary strategies such as MAXIMUM USABLE RANGE and DIFFERENTIAL SLOWING. It does not include what might be considered supporting or polishing strategies such as Avoid Interior Corners.

All of the agents solved each problem in a few seconds or less. For this reason, there was no need to model any of the satisficing strategies. This gave the computer an advantage in any task that required the agent to make precise measurements – humans are better at identifying key properties of the problem but, once identified, properties such as the number of path squares inside a tower's range can be calculated more precisely by a computer. While human methods, constraints and opportunities for satisficing are intrinsically interesting, since our goal was to create an optimal agent, we saw no reason to design the agents described here to not to take advantage of their inherent superiority in precision.

9.3.1 Advanced Maximum Usable Range – The Obvious Strategy

As mentioned in Section 9.2.3 MAXIMUM USABLE RANGE – A Purely Spatial Strategy, we showed the GopherTD task to a number of people (primarily academics at various AI conferences) who were not part of the study and they immediately stated that the solution to each problem was to place the towers where they covered the largest number of path tiles. We called this the MAXIMUM USABLE RANGE strategy. None of the subjects in any of the experiments used this problem solving approach but many used it as the base for a better problem solving style. We refer to this set of strategies as the ADVANCED MAXIMUM USABLE RANGE approach.

In the MAXIMUM USABLE RANGE strategy, all towers are placed where they will cover the largest number of path tiles. ADVANCED MAXIMUM USABLE RANGE differs in a few ways:

- **Traffic Volume**. The number of physical tiles in a tower's range is correlated with, but not identical to, the amount of time creeps spend in the tower's range. Traffic volume accounts for the situation when more than one line of creeps traverse a single path tile and where creeps in a given line traverse a given tile more than once. See STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME for more information.
- **Slow In Range**. The SLOW IN RANGE strategy says to place slowing towers near the heavy offense towers so that they can support the offense towers. By slowing the creeps that are in the heavy towers' range, the heavy towers have more time to attack the creeps. Although we normally refer to the ADVANCED MAXIMUM USABLE RANGE approach as a spatial approach since the core MAXIMUM USABLE RANGE strategy is spatial, SLOW IN RANGE is a temporal strategy. See STRATEGY TP1: SLOW IN RANGE for more information.
- **Heavy as Anchor**. In order to execute the SLOW IN RANGE strategy, the towers must be divided into roles and the groups must be prioritized. In this case, for

slowing towers to be placed around the heavy towers, the slowing towers must be placed after the heavy towers. The MAXIMUM USABLE RANGE strategy does not require tower roles, groups or ordering constraints between groups. Also unlike the MAXIMUM USABLE RANGE approach, our ADVANCED MAXIMUM USABLE RANGE Agent does not place slowing towers where they cover the highest number of path tiles. Instead, they are placed on the path adjacent tiles closest to the heavy offense towers.

It is worth explaining that last point. The strategies that select MAXIMUM USABLE RANGE and nearness work with ordinal properties. Attempting to prioritize one over the other often removes the contribution of the second strategy (e.g., if you prioritize nearness over usable range, the towers will always be next to the target tower even when positions with significantly more usable range are only a tile away). Attempting to use a weighted combination of two ordinal properties rather than prioritizing them requires normalizing the two values to a common currency (e.g., percent quality relative to peers) and determining the proper weights. These problems disappear when an ordinal property is combined with a boolean filter. In this case, combining nearness and path adjacency selects the free wall tile closest to the tower that also has good usable range.

The ADVANCED MAXIMUM USABLE RANGE approach was common among novices and many subjects with domain experience. Its performance is discussed in 11.2 Effects of Spatio-Temporal Independence

```
AdvancedMaximumUsableRangeAgent::getSolution(MapModel map)
   returns Solution
{
   SolutionRequest request = new SolutionRequest();
   request.MapModel = map
   request.towers   = getTowersToPlace()

   //--- Place offense towers where they get the most usable range
   SolutionRequest.groupStrategies.placementStrategies.Add(
      new PlaceTowerOnRelativePropertyStrategy(
         MapPropertyOrdinal.PathCellsInRange,
         StrategyOperator.Maximum))
   functionalGroup.tieBreaker = new RandomTieBreakerStrategy();

   //--- Place slowing towers next to the path...
   functionalGroup.placementStrategies.Add(
      newPlaceTowerOnAbsolutePropertyStrategy(
         MapPropertyAbsolute.PathAdjacent));

   //--- ...and close to the offense towers
   functionalGroup.placementStrategies.Add(
      new PlaceTowerNearGroupStrategy(0));
   functionalGroup.tieBreaker = new RandomTieBreakerStrategy();

   request.groupStrategies.tieBreaker = RandomTieBreakerStrategy
   return SolutionFactory.getSolution(request)
}
```

Listing 9.4. Advanced Maximum Usable Range Agent.

.

9.3.2 Differential Slowing – An Expert Strategy

The DIFFERENTIAL SLOWING family of temporal placement strategies attempts to separate the creeps so that the attack towers only have to worry about one group at a time. Figure 9.14 shows how subject 3E0200 used this strategy to earn a perfect score on the problem No Left Turns. This was one of the harder problems and those subjects who used a DIFFERENTIAL SLOWING strategy did well on this problem.

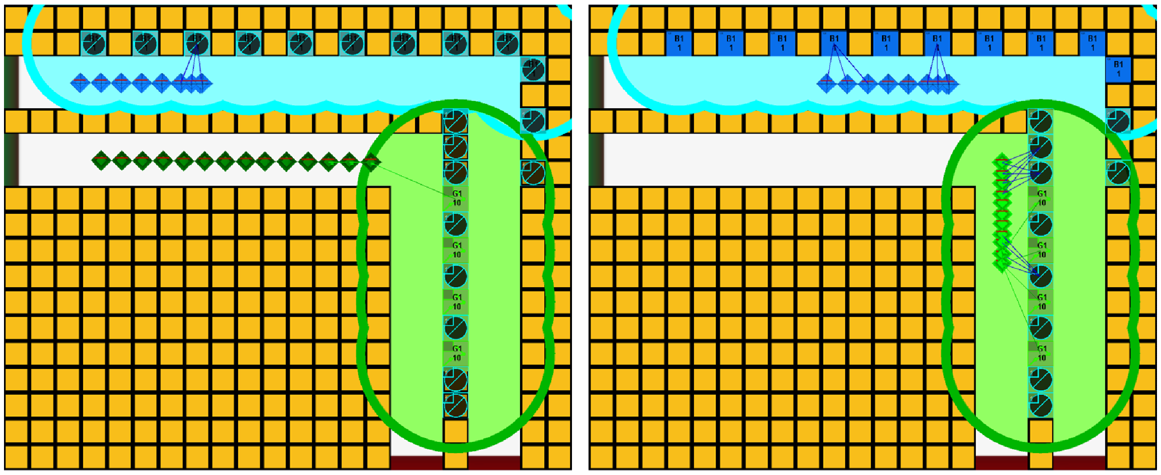


Figure 9.14. Subject 3E0200's Solution to No Left Turns in Action. LEFT: Creeps moving through the DIFFERENTIAL SLOWING zone (outer line) fall behind the creeps that do not (inner line). The creeps on the inner line enter the kill zone first. RIGHT: All of the creeps on the inner line will be in the kill zone before any creep on the outer line arrives.

The goal is to separate the two lines of creeps. There are two ways to do this. First, one line of creeps will naturally fall behind the other if they take a longer path. This is commonly the case when the creeps move around a corner – those on the outer path move further than the ones on the inside. Placing towers after a corner is STRATEGY TP8: EXPLOIT GEOMETRY.

The second way to separate the two lines is to place slowing towers where they can reach one line of creeps and not the other. These towers slow one group, allowing the other to pull forward. The most common way of doing this is Strategy TP9: Permanent Limited View DIFFERENTIAL SLOWING, although the DIFFERENTIAL SLOWING agent uses a modified version of the slightly more efficient STRATEGY TP10: PRE-KILL ZONE LIMITED VIEW DIFFERENTIAL SLOWING.

The EXPLOIT GEOMETRY and DIFFERENTIAL SLOWING strategies are based on the affordance of path synchronization gap, the amount of separation between two groups. Both strategies place attack towers after the position where the desired path synchronization gap is reached and thus both divide the map into two regions, the pre-

and post-separation regions. The DIFFERENTIAL SLOWING strategy places slowing towers in the pre-separation region. Both strategies place attack towers in the post-separation region. In Figure 9.15, there is a single corner. This corner introduces a path synchronization gap. The path leading up to that corner is in the pre-separation zone. By placing slowing towers in this region and making it so that they can only slow creeps on a single path, subject 3E0200 turns this region into a DIFFERENTIAL SLOWING zone (top right). All attack towers (bottom left) and the SLOW IN RANGE towers (bottom right) are in the post-separation region, forming a kill zone.

A DIFFERENTIAL SLOWING strategy only talks about how to separate a group, not what to do with it afterwards. The strategy by itself is not sufficient to solve a problem. It must be used in concert with other strategies. Our DIFFERENTIAL SLOWING agent also uses the SLOW IN RANGE strategy and a combination of strategies to rank positions for the Heavy Offense towers. The agent's strategy set:

- Role: Heavy Offense
    - Range After Differential Slowing Zone (i.e., the Differential Slowing towers)
    - Combination
        - Usable Range
        - Path Synchronization Gap
        - Coverage Balance
- Role: Slow In Range
    - Tower Touching Path
    - Tower Near Heavy Offense
- Role: Differential Slowing
    - First Attack Window Before Kill Zone (i.e., the Heavy Offense towers)
    - First Attack Window Intersect Path X
    - First Attack Window Not Intersect Path Y
    - First Attack Window Maximum Traffic Volume

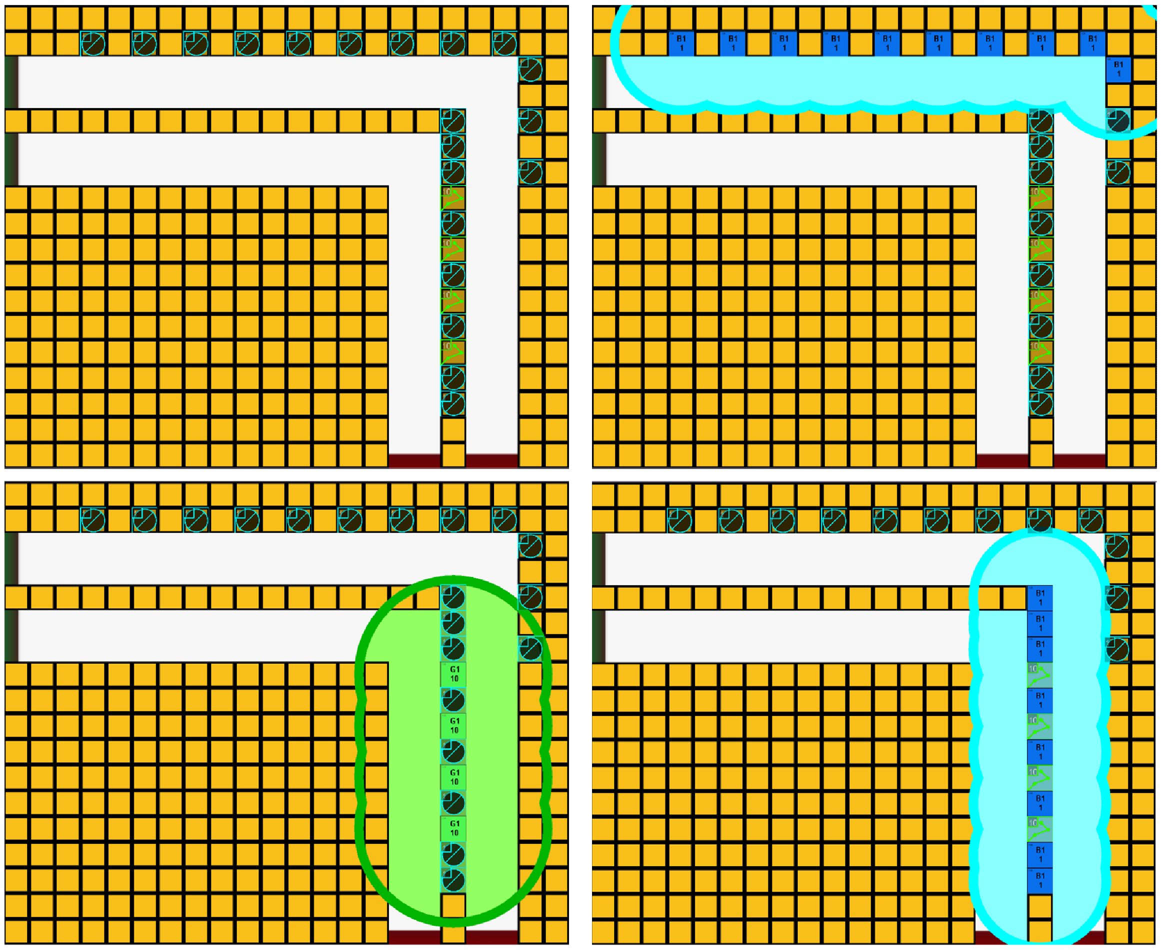


Figure 9.15. Subject 3E0200's Solution to No Left Turns. TOP LEFT: Towers are laid out to execute the DIFFERENTIAL SLOWING and SLOW IN RANGE strategies. TOP RIGHT: The DIFFERENTIAL SLOWING region. The slowing towers along the top of the map slow down the creeps on the outer / longer path but do not slow down the creeps on the inner line, causing the two groups to separate. BOTTOM LEFT: The kill zone. Attack towers are placed after the turn in the path, behind the DIFFERENTIAL SLOWING zone. BOTTOM RIGHT: The kill zone. Slowing towers are spread throughout the kill zone to delay the creeps, giving the attack towers in the kill zone more time to attack.

A combination of strategies is used to determine the best position for Heavy Offense towers. The actual formula used is:

```
utility = usableRange *
          (pathSynchronizationGap - coverageImbalanceGap)
```

Weights were determined by experimentation. The need for a coverage imbalance penalty was discovered during model creation; none of the subjects mentioned the issue in the experiments, perhaps because it was too obvious to mention. Path Coverage Ratio is the number of path tiles the tower covers for one path divided by the number of path tiles covered for the other path (smallest number is always divided by the larger number). The result is a number between 0 and 1 where 0 means only one path is covered, 1 means both paths are covered equally and 0.5 means one path is covered twice as much as the other.

```
pathCoverageRatio    = shortestPathLength / longestPathLength
coverageImbalanceGap = 3 – (3 * pathCoverageRatio)
```

The coverage imbalance measure was created after the agent consistently placed towers where they almost entirely ignored an entire line of creeps because doing so made their path synchronization scores better. An example is shown in Figure 9.16. This was a consistent problem across all maps. Exploiting the path synchronization gap is critical to many temporal strategies and is something that those humans who recognized it as a property seemed to have no problem doing. Initial positions picked by the agent were believed to be due to coding errors until we verified that the agent was, in fact, picking the positions with the highest path gap. As occurred many times in this work, it was hard for humans to make the mistakes the computer was making. Likewise, it was hard for us to explain how humans avoided these mistakes. We have not yet found an adequate model for this. We have crafted a serviceable solution by adding the coverage imbalance term and de-emphasizing the path gap. Path gap numbers are normally 2-6 while usable range is often in the 20s, so not weighting the terms means path gap makes a smaller contribution to the final score. This approach leads the agent to avoid the worst mistakes but it is a manually authored, problem-specific patch to solve a problem that we feel has a natural solution. For this reason, we believe that this problem deserves further study.

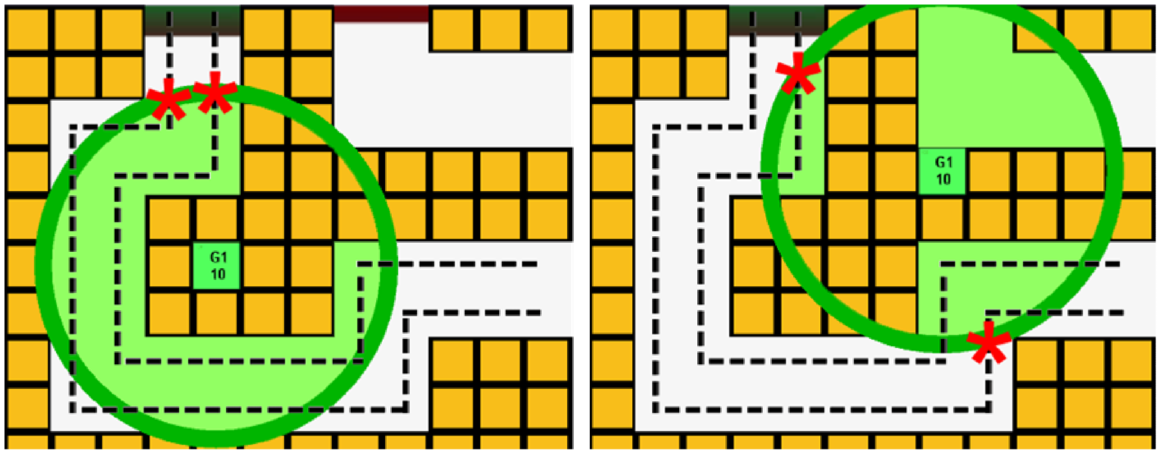


Figure 9.16. Coverage Imbalance Inflates Measurement of Path Gap. The path of two lines of creeps as they move around an area. Asterisks show the first time the line of creeps enters the tower's range. LEFT: Position maximizes the amount of path area it covers (usable range=25) but the lines enter at the same time (path gap=0). RIGHT: The tower is moved back to avoid attacking the outer line of creeps in its first attack window. The result is a large path gap (18) but less coverage (usable range=20). This reduced coverage comes entirely at the expense of the outer line, which is only in range for two path tiles. The tower effectively ignores one line of creeps in order to make its path synchronization score look good.

The agent's problem approach uses two map regions, a DIFFERENTIAL SLOWING zone and a kill zone. For this approach to work, the locations of these zones must be identified on the map. This turned out to be a significantly more complex task than was expected.

The DIFFERENTIAL SLOWING strategy requires part of the map be dedicated to a DIFFERENTIAL SLOWING zone and at least one other part of the map must be a kill zone. The DIFFERENTIAL SLOWING zone must be in a location where it is possible to slow one line of creeps without affecting the other. The kill zone should come after the DIFFERENTIAL SLOWING zone (i.e., the creeps should pass through the DIFFERENTIAL SLOWING zone before reaching the kill zone) and should be at a location where it can take advantage of the path synchronization gap caused by the DIFFERENTIAL SLOWING zone and map geometry.

Humans who used this strategy seemed to have little trouble partitioning the map into two pieces and finding locations for the two zones. As with many things that seemed simple when humans did them, attempting to implement this functionality in the computer agent turned out to be a non-trivial task.

On their own, identifying locations for the DIFFERENTIAL SLOWING and kill zones is not hard. As with most placement decisions in this domain, it can be made using only the data on the map. The task becomes more difficult when we add the ordering constraint between the two zones. The candidate positions for the kill zone must be after the DIFFERENTIAL SLOWING zone and vice versa. If the DIFFERENTIAL SLOWING zone is placed first and the best location for it is at the end of the map, there will be no room for the kill zone. If the kill zone is placed first and the best location for it is at the start of the map, the DIFFERENTIAL SLOWING zone will be of no use. Placing either zone requires reasoning about the opportunities left for the next one.

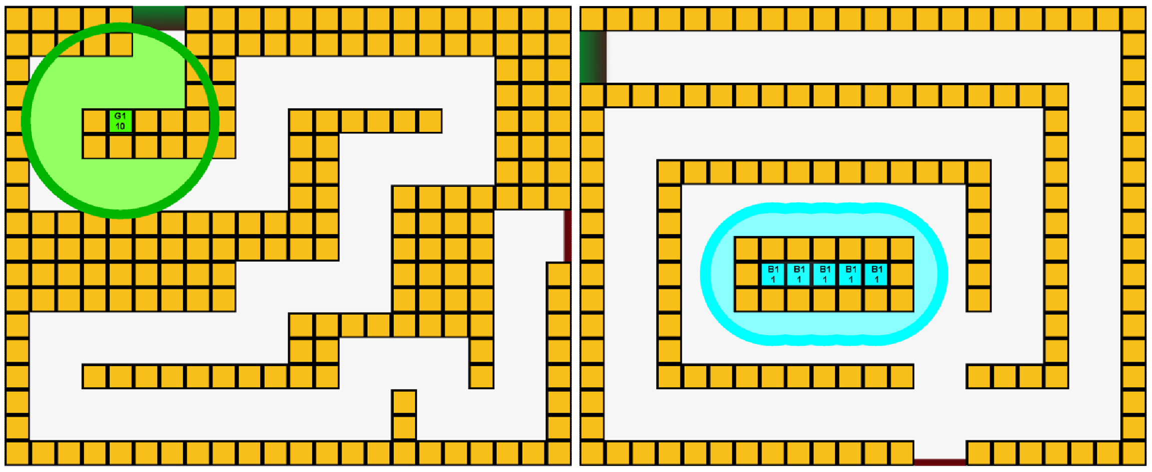


Figure 9.17. Tradeoffs in Differential Slowing. LEFT: Sometimes the best position for a kill zone is near the map entrance. Placing the kill zone too close to the entrance leaves no room for a DIFFERENTIAL SLOWING zone. RIGHT: Sometimes the best position for a DIFFERENTIAL SLOWING zone is near the map exit. Placing the DIFFERENTIAL SLOWING zone too close to the exit leaves no room for a kill zone.

The first task performed by our DIFFERENTIAL SLOWING agent is to break the map into three regions – required DIFFERENTIAL SLOWING, required kill zone and the area between that can be used by either zone.

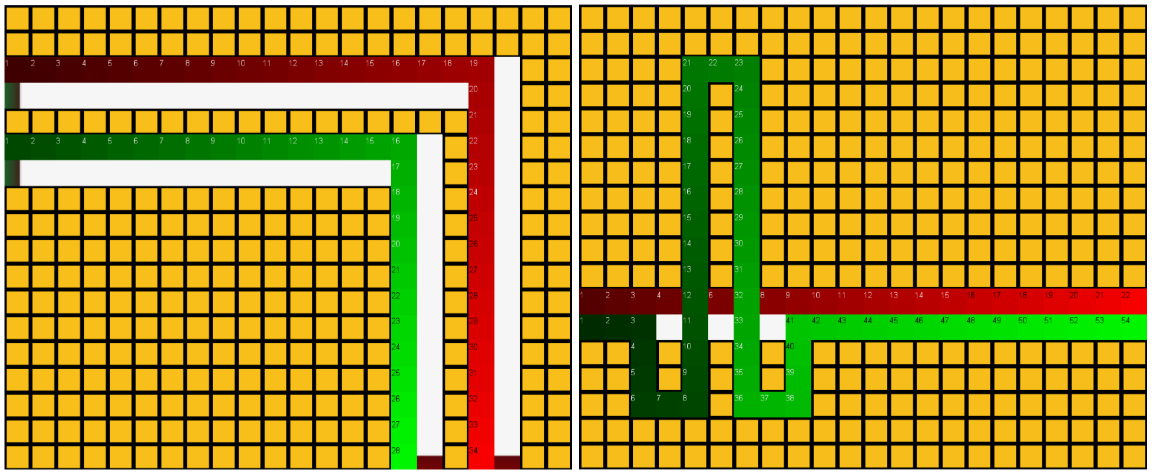


Figure 9.18. Path Through No Left Turns and Finger. LEFT: Map No Left Turns. Both paths enter at the left and move around the corner in the top right. RIGHT: Map The Finger. The top path (red) moves straight through while the bottom path (green) takes a longer, windier route.

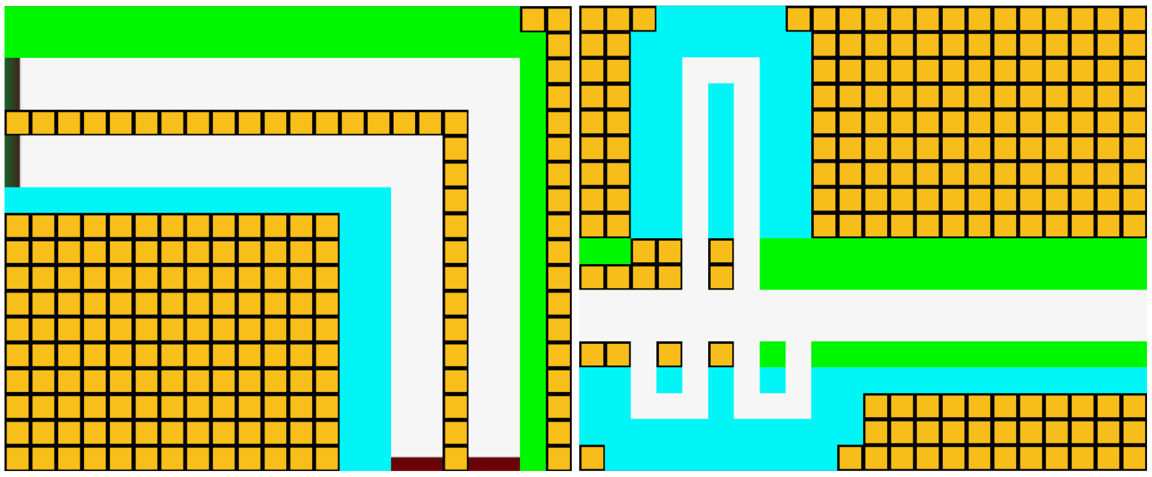


Figure 9.19. Single Path Overwatches. Positions where a Level 1 Blue1 slowing tower will slow creeps on one path but not the other. Green tiles indicate positions where towers will slow creeps on path 1. Blue positions indicate positions where towers will slow creeps on path 2.

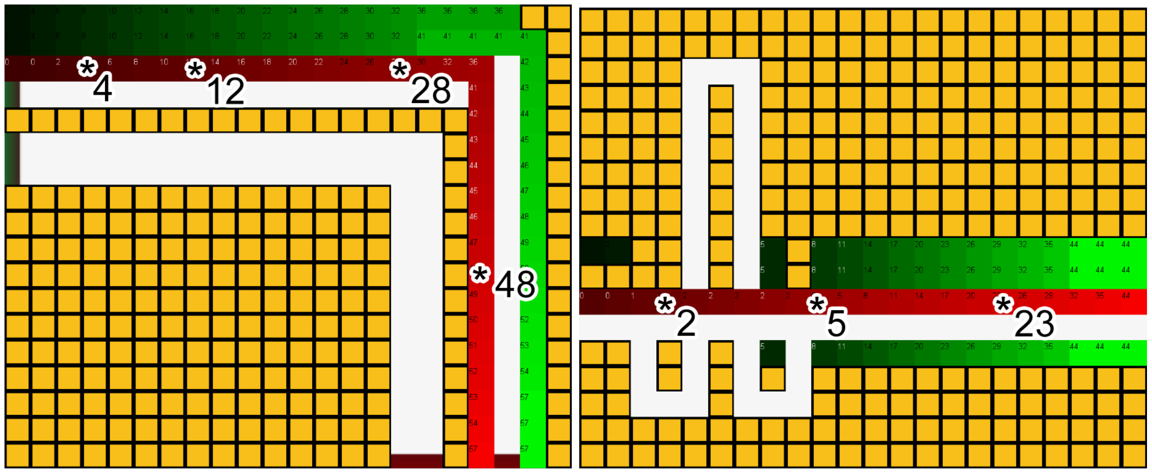


Figure 9.20. Differential Slowing Gain. The number of positions for DIFFERENTIAL SLOWING towers grows the further one goes on the map. In this example, the position holds a level 1 Blue1 tower whose first attack window only covers creeps on path 1. Numbers in the red path 1 tiles indicate the number of positions up to that point that are valid DIFFERENTIAL SLOWING positions. Green squares show the actual positions. Lighter colors indicate more towers. LEFT: The two lines of creeps are separated by a center island, making it easy for slowing towers above the island to only affect the creeps on path 1. RIGHT: There are very few positions at the start of the map that can slow path 1 and not path 2. This jumps dramatically in the second half of the map.

DIFFERENTIAL SLOWING positions are those positions where a slowing tower is able to slow one line of creeps but not the other. The number of slowing towers to use varied from subject to subject. Tests of our DIFFERENTIAL SLOWING agent used the same towers human subjects were required to use in Experiment 2. Of the 20 level 1 Blue1 slowing towers available, eight were used for DIFFERENTIAL SLOWING.

The required DIFFERENTIAL SLOWING region is the smallest region, starting at the map entrance, that contains the required number of DIFFERENTIAL SLOWING positions. Analysis is done using a sliding window along the path. A path tile is selected and the region between the starting path tile and the current one is searched. For each of these positions, if the desired slowing tower placed at that position covers the specified path, does not cover any other path and its range exit is before or on the current path tile, it is counted as being in that region and the number of candidate positions in that region is incremented.

The region end marker is then moved to the next path tile and the process is repeated. This continues until the specified number of positions is found. In our agent's case, the required DIFFERENTIAL SLOWING region ends once there are 8 DIFFERENTIAL SLOWING positions in the region between the map entrance and the current path marker.

All slowing towers in a DIFFERENTIAL SLOWING zone must slow the same line. As there are two lines of creeps, this sliding window analysis must be done twice, once relative to each path. The agent receives an instance of `EvaluationOfRegionSplitForDifferentialSlowing` for each path. This object contains information used by the agent to determine which path to target when applying the DIFFERENTIAL SLOWING strategy.

It is possible that the position that gives an offense tower the MAXIMUM USABLE RANGE is inside the required DIFFERENTIAL SLOWING region. Since offense towers must be placed after this region, the agent must be willing to sacrifice some usable range in order to use the DIFFERENTIAL SLOWING strategy. How much it is willing to lose is the Acceptable MAXIMUM USABLE RANGE Loss Percentage. For our agent, this number is hard coded to 45%.

The required kill zone region is the smallest region, ending at the map exit, that contains at least one position that is at or above the Acceptable MAXIMUM USABLE RANGE Loss Percentage. Analysis is done using a sliding window along the path. A path tile is selected and the region between the current path tile and the path end is searched. For each of these positions, if the desired tower placed at that position has a usable range that is no more than 45% less than the usable range of the tower at the best position on the map and has a range entrance that is on or after the current path tile, it is counted as being in that region and the number of candidate positions in that region is incremented. The region end marker is then moved to the next path tile, shrinking the region, and the process is repeated. This continues as long as there is at least one valid position in the region. In Figure 9.21, right, the MAXIMUM USABLE RANGE of a level 10 Green3 placed anywhere on the map is 29. Since the agent is willing to lose up to 45% (13 path tiles) of this amount, the required kill zone region will start at the map entrance and shrink until there are no longer any positions that give the tower a usable range of 16 or higher. The starting point of that region will be the path tile immediately previous to that one.

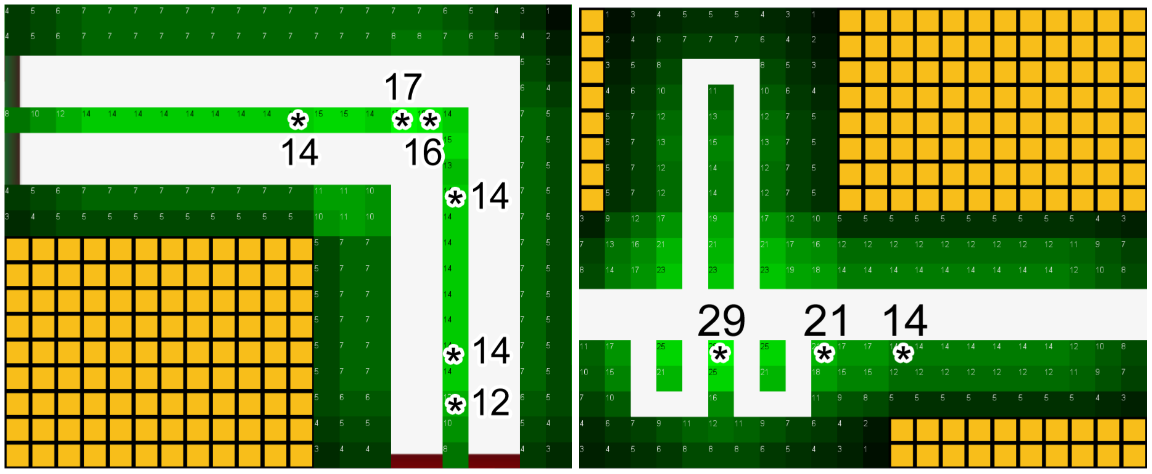


Figure 9.21. Usable Range Loss. The usable range of a level 10 Green3 tower at each position. Lighter colors have more range. LEFT: Most positions on the center island give a usable range of 14. Towers on the corner cover a little more area. 17 is the most usable range the tower can cover on this map. RIGHT: Towers placed in the center have a larger usable range because the bottom line of creeps circle those positions. Positions on the right side of the map cover half as much usable range. 29 is the most usable range the tower can cover on this map.

The third region is the area between the first and last regions, outside of the required DIFFERENTIAL SLOWING region and required kill zone region. It is up to the agent to decide how to allocate that space. We describe our agent's approach in a moment.

The number of DIFFERENTIAL SLOWING towers before and offense towers after each point on the path are calculated by the sliding window analysis. Each time one of the values changes (i.e., the starred yellow path tiles in Figure 9.22), that path tile is added to the list of value transition points. The value transition points are the candidates for region split points.

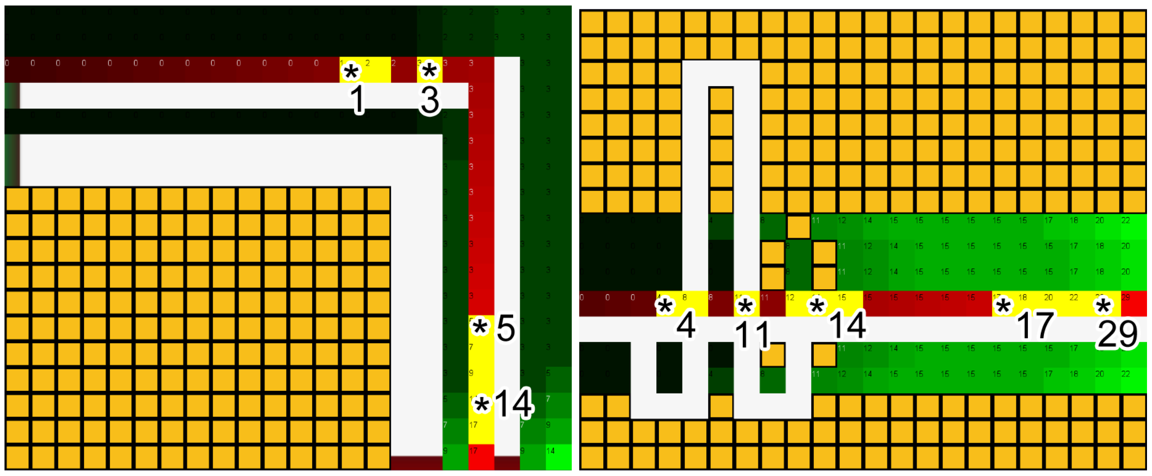


Figure 9.22. Maximum Usable Range Loss. Numbers in the red path 1 tiles indicate the potential amount of usable range a level 10 Green3 tower loses if it is placed after that position. LEFT: The global MAXIMUM USABLE RANGE on this map is 17, near the corner. The majority of center island positions give a range of 14. Positions after the corner lose a potential 3 usable range. Positions near the exit often give little usable range. If the tower were placed so that its range entrance was at or after the path cell marked *14, it would cover 14 fewer path cells than if it had been placed at the globally optimal position. RIGHT: The global MAXIMUM USABLE RANGE on this map is 29, center of map. Positions to the right of this offer considerably smaller usable ranges.

The DIFFERENTIAL SLOWING strategy requires the kill zone to come after the DIFFERENTIAL SLOWING zone. The required regions are the smallest regions necessary to implement the respective roles. If the end point of the required DIFFERENTIAL SLOWING region comes after the starting point of the required kill zone (Figure 5.2, right), the two overlap, meaning that there are no acceptable offensive positions after the DIFFERENTIAL SLOWING zone. In this situation, the DIFFERENTIAL SLOWING strategy cannot be used on that path on that map and the agent must either try it on the other path or switch to another strategy.

Identifying the value transition points allows us to identify the required region start and end points. Once these are identified, the actual location of the DIFFERENTIAL SLOWING and kill zones must be determined. The candidates for each of these include both their respective required region and the unclaimed positions in the middle region.

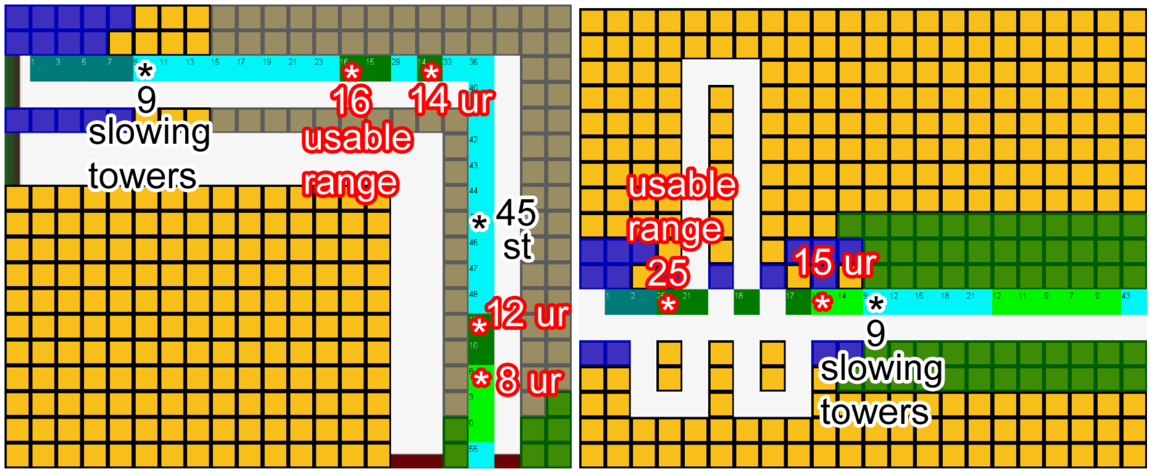


Figure 9.23. Value Transition Points. Required DIFFERENTIAL SLOWING region in blue, Required Kill Zone in green, middle region in gray. Path tiles colored blue for number of DIFFERENTIAL SLOWING towers, green for usable range. LEFT: The required number of DIFFERENTIAL SLOWING towers, 8, is reached early in the map. The agent requires an offense position that offers a usable range of 10 or more, which exists near the exit. The majority of the map is open. RIGHT: The required number of DIFFERENTIAL SLOWING towers, 9, is reached just after the center of the map. The agent requires an offense position that offers a usable range of 16 or more, the last of which is a few path tiles earlier than the end of the required DIFFERENTIAL SLOWING range. Because the two required regions overlap (i.e., the kill zone does not come after the DIFFERENTIAL SLOWING zone), the DIFFERENTIAL SLOWING strategy cannot be used on this path on this map.

Our agent places the kill zone first. The candidate set includes all positions in the middle and required kill zone regions. As mentioned earlier, the agent evaluates each of these positions for usable range, path synchronization gap and coverage balance.

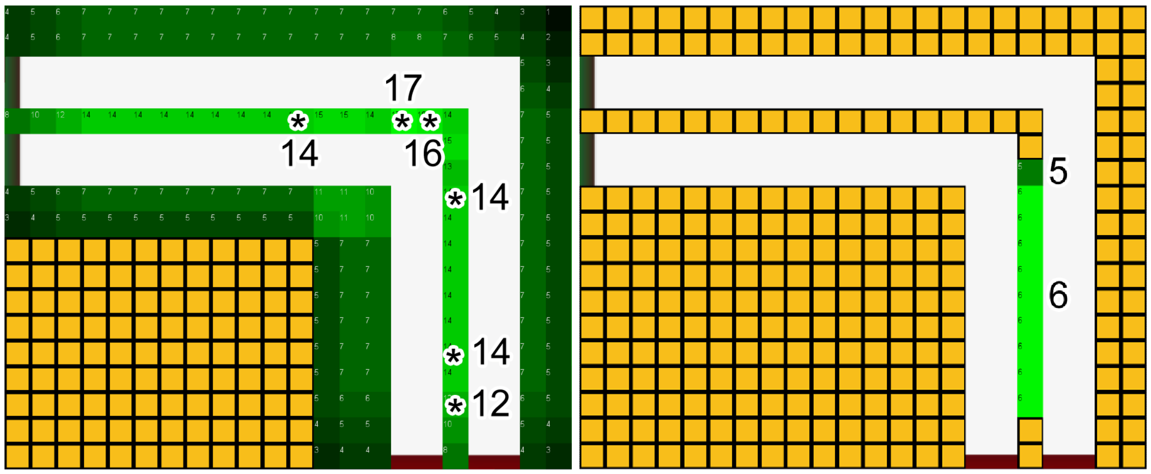


Figure 9.24. Usable Range and Path Gap. LEFT: Usable range for a level 10 Green3 tower. Light color means larger range. Uncolored walls means a tower placed there will not cover any path tiles. RIGHT: Path synchronization gap. Lighter color means higher gap. Uncolored means there is no gap. There are no positions with a path gap before the corner as it is the corner that causes the outer line of creeps to fall behind the inner line, introducing a path synchronization gap.

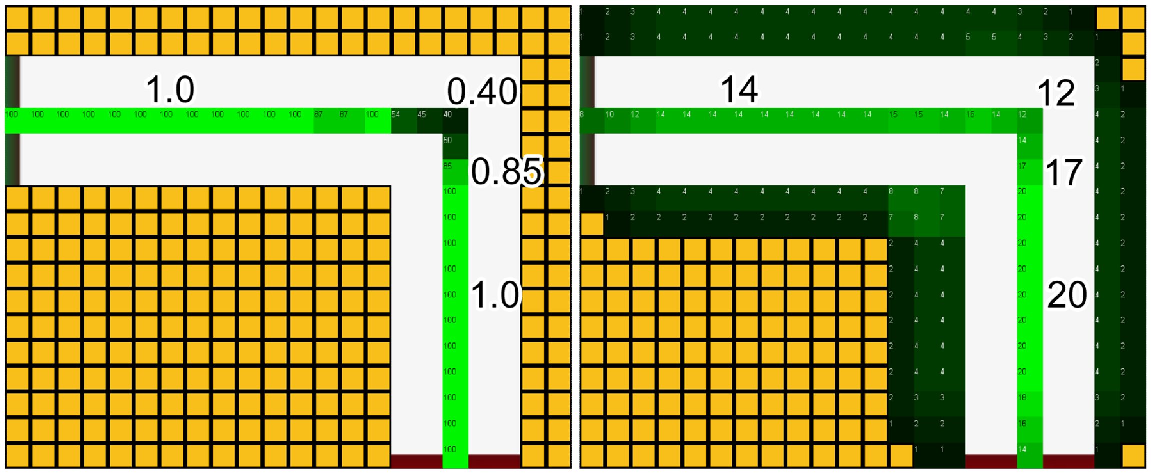


Figure 9.25. Coverage Balance and Utility. LEFT: Coverage balance. Lighter color means the tower covers an equal number of path tiles in path 1 and path 2, darker color means the tower covers one path more than the other. RIGHT: Utility map. Weighted sums of usable range, path synchronization gap and coverage balance. The top and bottom positions have similar usable range and coverage balance but the bottom positions offer a path synchronization gap that the top ones do not have. The corner offers a greater usable range but that range is heavily biased towards the outer path.

Performance tests were conducted under the constraints from Experiment 2, in which the problem solver is given four level 10 Green3 heavy offense towers and 20 level 1 Blue1 slowing towers. In this situation, the offensive part of the kill zone is completed once the agent places four Green3 towers. The kill zone will also contain SLOW IN RANGE towers (the 12 slowing towers not used in the DIFFERENTIAL SLOWING area) but because it is okay for these to straddle the border of the DIFFERENTIAL SLOWING and kill zones, we will place them last.

The location of the heavy offense towers is stored in a `GroupNeighborhood`. The boundaries of this object mark the actual boundaries of the kill zone. The starting position of this zone will be the range entrance of the earliest offense tower in the group and will be somewhere between the end of the required DIFFERENTIAL SLOWING region and the start of the required kill zone region.

```
List<GridPoint> wallsInRegion =
    solution.map.getPositionOfWallsAfterPathCell(
       heavyTower, winningEvaluation.targetDSReachedAt, pathID);
PositionScores scores = SAQS.getScoresForUsableRangeAndPathGap(
    heavyTower, solution.map, wallsInRegion);
scores = scores.getTop(numberOfHeavyOffenseTowers)
foreach (TowerLog tower in scores)
{
    killzone.add(heavyTower, tower.getPosition());
}
```

Listing 9.5. Place Heavy Offense Towers in the Kill Zone.

Once this position is known, we can allocate towers to the DIFFERENTIAL SLOWING zone. The candidate set of positions contains all positions between the map entrance and the start of the kill zone where a slowing tower will slow the desired line of creeps but not the other line. From this set of positions the agent selects those that give the slowing towers the MAXIMUM USABLE RANGE. It is likely a better strategy to spread the slowing towers out, spacing them equally along the path, but at the time this document is being written we have not yet implemented the code to place chains of towers.

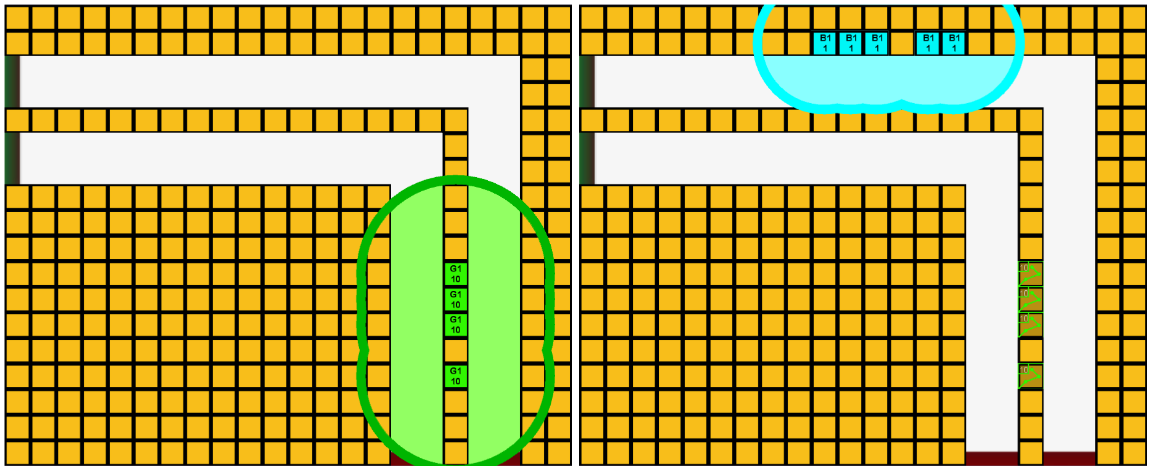


Figure 9.26. Zones Selected by the DIFFERENTIAL SLOWING **Agent**. LEFT: The kill zone. RIGHT: The DIFFERENTIAL SLOWING zone.

```
MapCellMetaData cell = killzone.getFirstPathCell(pathID);
int endOfDSRegion = cell.globalStepNumber;
wallsInRegion =
        solution.map.getPositionOfFreeAttackWindowsBeforePathCell(
            slowingTower, endOfDSRegion, pathID);
List<GridPoint> dsPositions =
    SAQS.getPositionsForSlowingOneLineAttackWindow(
        slowingTower,
        solution.map, pathID, wallsInRegion);
scores = SAQS.getScoresForPathCellsInRange(
    slowingTower, solution.map, dsPositions);
scores = scores.getTop(numberOfDifferentialSlowingTowers)
foreach (TowerLog tower in scores)
{
    dsZone.add(slowingTower, tower.getPosition());
}
```

Listing 9.6. Place Slowing Towers in the DIFFERENTIAL SLOWING **Zone**.

Once the DIFFERENTIAL SLOWING towers are placed, the remaining slowing towers are placed around the heavy offense towers to execute the SLOW IN RANGE strategy. As with the ADVANCED MAXIMUM USABLE RANGE agent, SLOW IN RANGE towers are placed on the path adjacent cells closest to the towers in the heavy offense group.

Figure 9.27 shows the final solution generated by the DIFFERENTIAL SLOWING agent and the solution generated by subject 3E0200. Both solutions placed the DIFFERENTIAL SLOWING zone before the corner and the kill zone after and both chose to slow the outer path. They differ in how slowing towers were allocated between the DIFFERENTIAL SLOWING and kill zones. This was a design decision – the agent is hard coded to place a fixed number of agents in the DIFFERENTIAL SLOWING zone. It is not known how sensitive this number is to context and how an agent would determine the number to use for a given problem. This topic is left for future work.

```
PlaceTowerOnAbsolutePropertyStrategy pas =
   new PlaceTowerOnAbsolutePropertyStrategy(
      MapPropertyAbsolute.PathAdjacent);
List<GridPoint> freeWallCells = pas.getPositions(
   solution.map, solution.map.getPositionOfFreeWallCells());
PlaceTowerNearGroupStrategy pngs =
   new PlaceTowerNearGroupStrategy(0);
RandomTieBreakerStrategy tieBreaker = new RandomTieBreakerStrategy();
for (int i = 0; i < numSIRTowers; i++)
{
   List<GridPoint> closestWalls = pngs.getPositions(slowingTower,
      solution.map, freeWallCells, killzone.getTowerPositions());
   GridPoint position = tieBreaker.getPosition(closestWalls);
   solution.addTower(slowingTower.toTowerLog(), position);
   freeWallCells.Remove(position);
}
```

Listing 9.7. Place Slow In Range Towers in the Kill Zone.

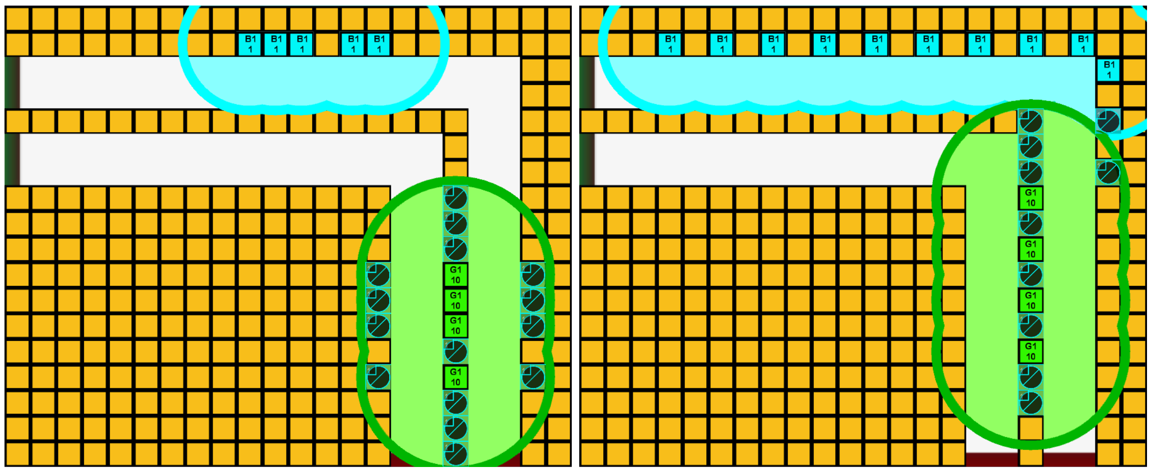


Figure 9.27. Final Solutions for Human and Differential Slowing Agent Compared. LEFT: The solution generated by the DIFFERENTIAL SLOWING agent. RIGHT: The solution generated by subject 3E0200.

One large difference between the two solutions is in tower distribution. Subject 3E0200 spaced their heavy offense towers every other tile so that slowing towers could be placed between them, keeping the creeps consistently slow without gaps. The DIFFERENTIAL SLOWING zone is stretched to run the full length of the path up to the kill zone. Again, the subject places their towers every other tile. The GopherTD system does not currently support this type of chain placement/density distribution strategy.

The DIFFERENTIAL SLOWING approach was common among subjects with domain experience. No domain novice used it. Its performance is discussed in 11.2 Effects of Spatio-Temporal Independence.

Chapter 10

Experiment 2

Strategy Performance

Experiment 1 looked at the breadth of strategies and representations for solving tower defense puzzles. Given the amount of variability in the approaches that subjects used – scores depended not just on tower placement strategies but also tower selection strategies and subject experience – there was no way to reliably measure the quality of the observed strategies. This was sufficient to build computer agents that could simulate specific humans but did not help in the design of better-than-human problem solving agents. The goal of Experiment 2 was to measure the quality of the observed strategies.

10.1 Setup and Constraints of Experiment 2

Several steps were taken to make inter-subject comparisons meaningful:

- **Consistent Towers**. All subjects were required to use the same towers. The choice of towers was determined in two ways. First, a series of simulations were run using different sets of towers. Second, we looked at the towers used by the highest scoring subjects in earlier experiments. Both sources agreed on the optimal set of towers: four level 10 Green3 offense towers and 20 level 1 Blue1 slowing towers.
- **Controlled Experience**. 20 subjects participated, half with significant domain experience and half with none. This allowed us to measure the contribution of domain experience to performance.
- **Consistent Training**. All subjects trained on the same maps in the same order the same number of times.
- **Known Problems**. Unlike previous experiments, training was performed on all maps. The testing phase did not use any novel problems and problem performance did not depend on transfer ability.

The goal of Experiment 2 was to determine the effectiveness of the spatio-temporal placement strategies. Because many strategies depended on the affordances of a given tower, restricting the allowed towers had the effect of reducing the number of strategies available to the subject. For example, subjects did not have access to the red towers, which are the only towers with a range large enough to cover more than two path corridors, and therefore had little incentive to learn strategies based on maximizing the number of path corridors covered. Likewise, subjects did not have access to the brown freezing towers and therefore could not effectively execute STRATEGY TP2: SWAMP. Reducing the number of strategies used was necessary to obtain a large enough sample size to accurately measure the performance of the strategies.

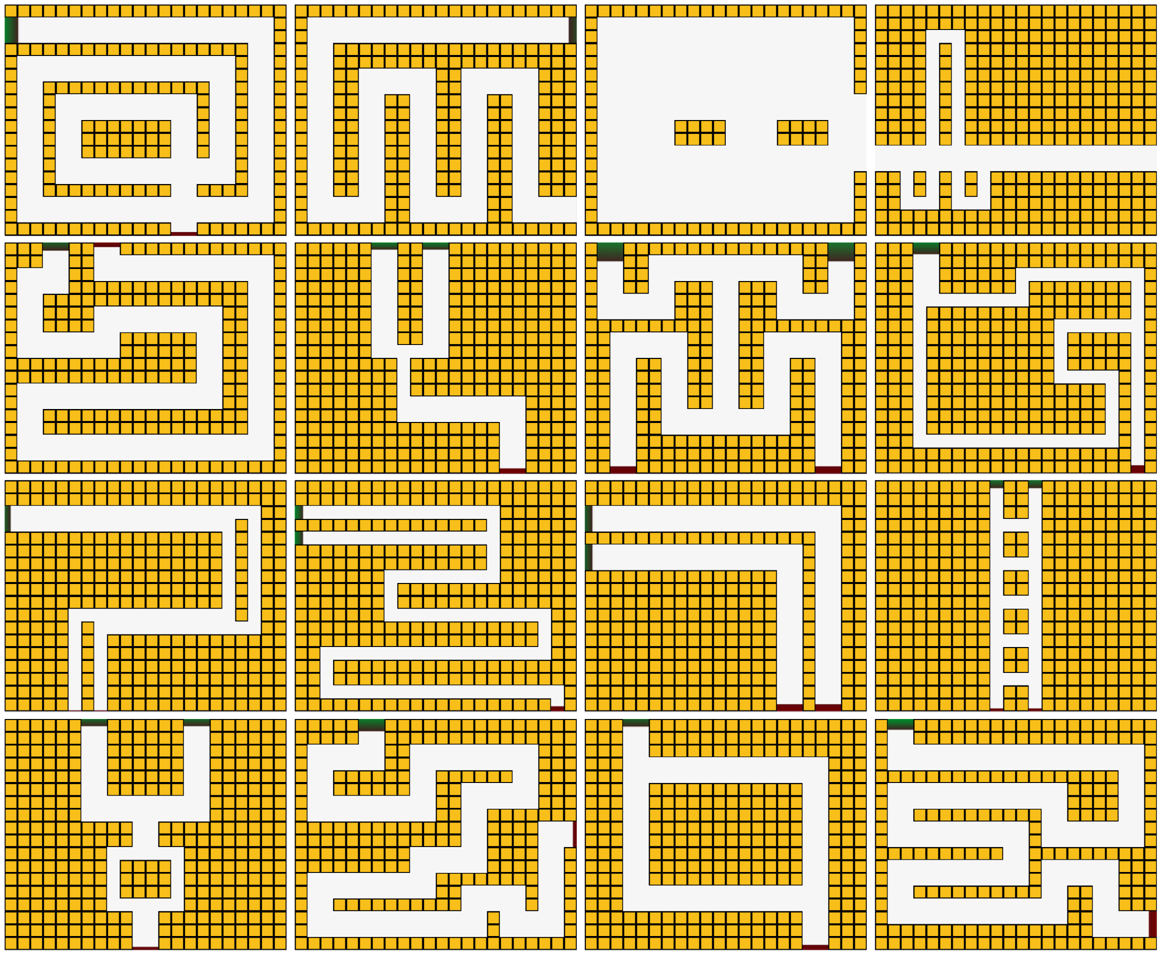


Figure 10.1. Experiment 2 Maps.

Category	**Count**	**Male**	**Female**
Experienced	10	10	0
Novice	10	5	5
Total	20	15	5

Table 10.1. Demographics, Experiment 2.

16 maps were used. VectorTD only contained 14 maps. Hoping to repeat the insight effect seen with the map Void, two additional maps were created with the intention of strongly suggesting strategies. Finger (Figure 10.1, map 4) was similar to Void but with the relevant affordance exaggerated in hopes of suggesting the DIFFERENTIAL SLOWING strategies to more subjects. No Support (Figure 10.1, map 3) was intended to teach

STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION. From the observed results, it does not appear that either was successful in prompting the discovery of new strategies.

For this experiment, 20 subjects were tested, 10 with prior experience with tower defense puzzles, 10 without. None of the subjects had participated in the first two experiments. Subjects trained on all test maps so that the tests measured the subject's ability to execute a strategy rather than their ability to transfer it to a novel problem.

Training	**Min**	**Mean**	**Max**
Trials	18	37.10	50
Minutes	99	160.32	246

Table 10.2. Time Spent Training, Experiment 2.

Training. Training in Experiment 2 differed from the earlier experiments in several ways. All 16 testing maps were used in training. Subjects were asked to practice each map three times (i.e., 48 training trials). Subjects did not control the order of the training maps. Each map was presented in the order it would be seen in during testing. Subjects made three attempts on a map and then moved to the next one.

Subjects in Experiment 2 trained twice as much as subjects in earlier experiments did, solving an average of 37 problems (vs. 17) in 160 minutes (vs. 86).

Difficulty	**Maps**	**Count**	**Difficulty Sequence**
Low	1, 10, 16	3	`E,M,M,M, M,H,M,M,`
Moderate	2-5, 7-8, 12, 14	8	`H,E,H,M, H,M,H,E`
High	6, 9, 11, 13, 15	5	

Table 10.3. Experiment 2 Map Difficulty.

Testing. 16 maps were used in the testing phase. All maps had been seen previously in training.

As before, no break was given between training and testing.

10.2 Measured Strategies

Most, although not all, subjects used either the ADVANCED MAXIMUM USABLE RANGE (AMUR) strategy or a DIFFERENTIAL SLOWING (DS) strategy. AMUR is a spatial proxying strategy in which space and time are treated as interchangeable and space is maximized. DS treats space and time as independent and seeks to maximize time. As has been argued throughout this paper, humans solve complex problems by applying a set of simple, focused strategies rather than a single comprehensive one. Correspondingly, each subject created a unique solution for each problem. Because accounting for each solution would make the data set too sparse to perform statistical analysis and because we believe that the choice of AMUR or DS was responsible for the largest part of the differences in solution performance, we chose to group subjects into two groups based on their use of AMUR or DS. Subjects who used neither strategy were omitted from analyses that involve grouping by strategy but not from analyses based on grouping by experience level. Because some subjects were inconsistent in their strategy use (i.e., participants did not necessarily apply a given strategy to all maps), the number of subjects reported for each strategy group varied by map.

The components of these strategies are explained in detail in Chapter 8 Strategies but we summarize the important points here.

AMUR is not technically a strategy but a pair of strategies, MAXIMUM USABLE RANGE and SLOW IN RANGE. MAXIMUM USABLE RANGE refers to any of the strategies in Section 8.1.1 To Increase Coverage Area, where the goal is to place the tower such that its range covers as much of the path area (i.e., the area where the creeps will be) as possible. It attempts to maximize the amount of space covered, making it a spatial strategy.

STRATEGY TP1 SLOW IN RANGE says to place slowing towers around the offense towers so that the creeps spend more time in the offense towers' range. This strategy explicitly maximizes time, not space, meaning that AMUR is technically a hybrid spatio-temporal strategy, but one that focuses more heavily on space.

It is tempting to think that subjects using the same strategy (or, in this case, pair of strategies) would get the same score. We observed three reasons for why this didn't happen. The first is that subjects use more than these two strategies, and the additional strategies contribute to performance. Many of these are optimizing strategies that subjects create during testing. An example is STRATEGY E1 AVOID SINGLE-SIDED INTERIOR CORNERS. This strategy eliminates from consideration a specific class of suboptimal positions, reducing a certain type of waste that many subjects don't immediately notice. This also explains why some subjects felt that their performance increased with experience (as the maps are non-comparable, there is no way to verify this, but our intuition is that many of these subjects are correct).

A second reason why subjects using the same strategy achieved different scores is that MAXIMUM USABLE RANGE is not a strategy but a family of strategies. While subjects attempted to maximize a tower's usable range, the definition of usable range varied between subjects. Subjects also varied in how they measured usable range. Details are provided in Section 8.1.1 To Increase Coverage Area. While all of these strategies attempt to maximize space, they differ in effectiveness. Note that seven of the spatial measurement strategies are satisficing strategies. In this domain, a computer agent can quickly determine which position(s) offer the MAXIMUM USABLE RANGE. As measured by a combination of answer quality and run-time performance, it is believed that these specific satisficing strategies used by humans underperform the optimal ones used by the computer under the current conditions. It is possible that the satisficing strategies would outperform the computer under other conditions, such as significantly larger maps, where a human's ability to visually pick out features such as U-turns might be significantly faster than measuring the value at every position as a non-satisficing computer would do.

The third reason why some subjects outperformed others using the same strategy is simple execution failure. Subjects sometimes didn't notice certain parts of the map and in at least one instance a subject conflated his strategies, accidentally applying a strategy meant for towers in one role to a tower in a different role. Even under laboratory testing conditions, subjects were sometimes distracted and as a result made mistakes.

As with MAXIMUM USABLE RANGE, DIFFERENTIAL SLOWING is not a single strategy, it is a family of strategies. Details are provided in Section 8.2.2 To combat attack window decay Strategy Family: Differential Slowing, strategies TP9-TP15. We also include the related STRATEGY TP8 EXPLOIT GEOMETRY. Just as all of the MAXIMUM USABLE RANGE strategies involved maximizing space, all of these strategies involve separating the two lines of creeps. A path synchronization gap between the two lines of creeps is created and/or exploited, with one line falling behind the other. In the best case, the two lines are fully separated, doubling the offensive towers' activation time as they can use the normal amount of time they get to attack the first group of creeps and then use it again to attack the later group. It is often the case that the place where the two lines of creeps are most separated is not the place where the tower would achieve its MAXIMUM USABLE RANGE. In these cases, the problem solver obtains more time by giving up space.

In summary, AMUR assumes that space and time are perfectly correlated and therefore the best way to increase a tower's activation time is to increase the tower's usable range. Differential Slowing assumes that space and time are independent and that the best way to increase a tower's activation time is sometimes to pick positions that decrease the tower's usable range. We therefore consider AMUR to be a predominantly spatial strategy and Differential Slowing to be a predominantly temporal strategy.

Chapter 11

Results of Experiment 2

In this chapter we are interested in which factors impact the problem solver's performance. We are specifically interested in the following factors:

- Domain Experience
    - Does experience with a related domain improve performance?
    - Does experience with a related domain change the strategies used?
    - Does experience with a related domain decrease time-space conflation?
    - Does experience with a related domain improve how well a strategy is executed?
- Space-Time Conflation
    - Does treating space as a proxy for time outperform baseline strategies?
    - Does treating space and time as independent outperform treating space as a proxy for time?
- Human Ability vs. Computer
    - How well do our strategy agents do relative to baseline strategies?
    - How well do our strategy agents do relative to humans in different experience groups?

- How well do our strategy agents do relative to humans using the same strategies?

A detailed analysis of the data is presented in Appendix D Detailed Statistical Analysis of Experiment 2. Preliminary results were published in (Wetzel et al. 2012). We focus here on the lessons to draw from the data.

11.1 Effects of Domain Experience

One of the questions we wished to answer was whether experience on conceptually similar tasks would profitably transfer to a novel domain. In other words, do people who have played other tower defense games do better on new tower defense games than people who have never played one?

11.1.1 Impact of Experience on Performance

The data indicates that experience with a related domain helps performance, but not to a large degree. Subjects with experience did much better on one map and modestly better on three maps (Table 11.1). The effect of experience on the remaining 12 maps was small or statistically insignificant.

A 2 (experience level) x 16 (map) mixed factor ANOVA analysis showed:

- a main effect of map, $F(5.23, 94.15) = 98.11$, $MSE = 19.92$, $p < .001$
- a marginal map x group interaction, $F(5.23, 94.15) = 2.06$, $MSE = 19.92$, $p =.074$
- a main effect of group, $F(1, 18) = 4.75$, $MSE = 24.55$, $p =.043$

Map	**Mean (SD)** Exper.	Novice	**Diff.**	**Comparison**	**Significance**
NoSupport	16.50 (4.01)	18.13 (1.19)	-1.63	Slight novice advantage	Nonsig.
RoundTheTwist	24.80 (3.14)	25.78 (2.22)	-0.98	Essentially no difference	Nonsig.
Finger	18.73 (1.80)	19.53 (1.18)	-0.80	Essentially no difference	Nonsig.
DoTheSplits	5.68 (1.57)	5.63 (1.07)	0.05	Essentially no difference	Nonsig.
Switchback	22.65 (3.03)	22.48 (2.51)	0.18	Essentially no difference	Nonsig.
Ladder	11.53 (1.27)	10.85 (0.93)	0.68	Essentially no difference	p = .19
2in1out	8.43 (1.68)	7.50 (0.65)	0.93	Essentially no difference	p = .12
SlimPickings	18.53 (3.29)	17.50 (4.01)	1.03	Slight experienced advantage	Nonsig.
SnakingPath	18.95 (4.81)	17.73 (2.00)	1.23	Slight experienced advantage	Nonsig.
Bottleneck	9.80 (2.20)	8.20 (0.86)	1.60	Slight experienced advantage	p = .046*
TheFrog	13.93 (2.81)	12.28 (1.73)	1.65	Slight experienced advantage	p = .13
NoLeftTurns	7.78 (3.74)	5.90 (1.36)	1.88	Slight experienced advantage	p = .15
Void	16.05 (3.75)	13.96 (3.15)	2.09	Modest experienced advantage	p = .195
UpAndDown	19.13 (3.55)	16.23 (4.56)	2.90	Modest experienced advantage	p = .13
Pathways	11.08 (3.57)	7.15 (1.61)	3.93	Modest experienced advantage	p = .005**
Elemental_ish	20.15 (3.99)	15.55 (4.57)	4.60	Large experienced advantage	p = .027*

Table 11.1. Average Score per Map, By Experience. Sorted by magnitude of experience advantage. SD = standard deviation; Exper. = experienced; * indicates p < .05, ** indicates p < .01.

Across all maps, experienced players (M = 15.23) outperformed novice (M = 14.02) players by a statistically significant amount. The magnitude, however, was not large – the

1.21 point gap represents a 4% advantage. Given that some maps were designed to be harder than others, it is not surprising to find that the main determinant of a subject's score was not experience but the map they were solving. Performance ranged from a mean of 5.65/28 (approximately 20%) to 25.28/28 (90%).

Mean		Experienced
Experienced	Novice	Advantage
15.23	14.02	4.3%

Table 11.2. Comparison of Subjects by Experience.

11.1.2 Impact of Experience on Strategy Selection

Of the 10 subjects with no previous domain experience, 70% used the spatial AMUR strategy set and 30% used other strategies (STRATEGY SP4: MAXIMUM PERCENT OF PATH, STRATEGY SP5: MAXIMUM NUMBER OF PASSES and STRATEGY TP16: KILL ZONE JUMPING). No domain novice used the temporal DS strategy set.

Of the 10 subjects with tower defense experience, 40% used the spatial AMUR strategy set, 50% used the temporal DS strategy set and 10% used another, more advanced, temporal strategy (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION).

While having domain experience did not guarantee that the subject used a strategy that explicitly maximized time, not having domain experience did seem to ensure that they did not. Domain novices maximized space, strongly suggesting that they used space as a proxy for time. With previous domain experience, some subjects used strategies that treated time and space as independent while others did not.

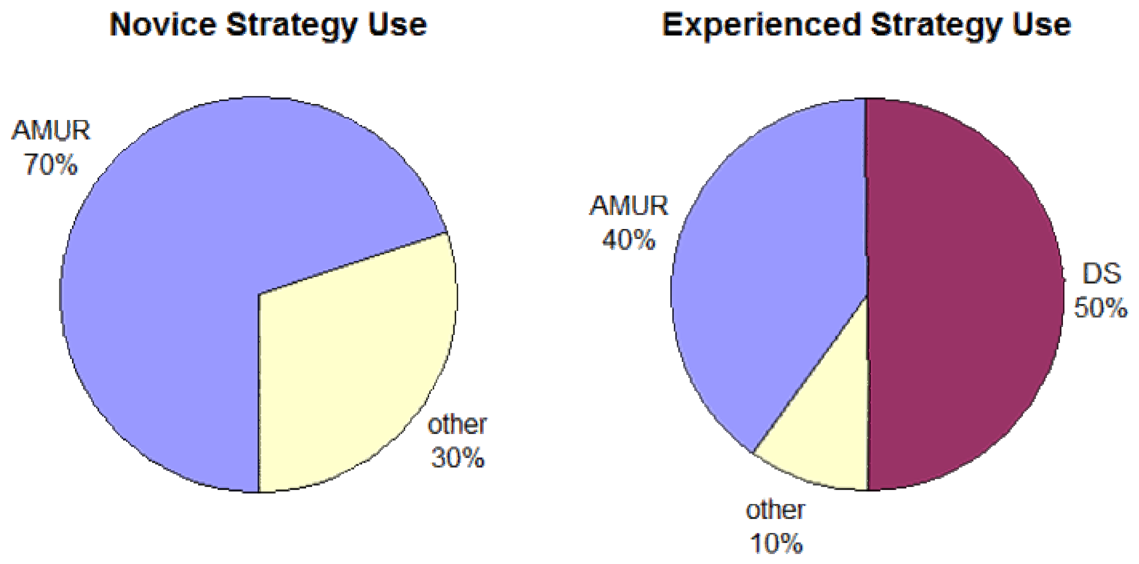


Figure 11.1. Subject Strategy Selection as a Function of Domain Experience. The spatial strategy Advanced Maximum Usable Range (AMUR) was used most often by subjects with little domain experience; the temporal strategy Differential Slowing (DS) was often (although not always) used by subjects with previous domain experience.

Combined with the findings of the previous section, it appears that experience with a related domain does not, on average, substantially improve performance but it has a strong probability of changing the strategies subjects use.

11.1.3 Impact of Experience on Time-Space Conflation

The previous section answers another question we had, does experience affect the degree of time-space conflation? 70% of domain novices used a strategy that maximized space rather than time. Only 10% used a strategy related to time (STRATEGY TP16: KILL ZONE JUMPING). In contrast, only 40% of the domain experienced group used strategies that maximized space while the remaining 60% used strategies that maximized time.

Subjects were divided into two groups based on whether they had any experience with tower defense puzzles. We did not account for degree of experience. It is possible that the 40% of subjects who used spatial proxying had minimal experience while those who used temporal strategies had more. The question is left for future work.

11.1.4 Impact of Experience on Strategy Execution

Does experience with a related domain improve how well a strategy is executed? In the case of the temporal DS strategy set, we cannot answer that question as no domain novice used it. Subjects in both the domain novice and experienced groups used the spatial AMUR strategy set so we focus on whether or not experience leads to better execution of the AMUR strategy.

The answer appears to be no. A 2 (experience level) x 16 (map) mixed factor ANOVA for only those humans who adopted an AMUR strategy showed:

- a main effect of map, F(4.29, 38.64) = 54.76, MSE = 22.80, $p < .001$
- no map x group interaction, F(4.29, 38.64) = 1.70, MSE = 22.80, p = .17
- no main effect of group, F(1, 9) < 1

For those humans who used the spatial AMUR strategy set, there was no statistically significant difference between the experienced players (M = 14.22) and the novice players (M = 13.74).

Mean Experienced AMUR	Novice AMUR	Experienced Advantage
14.22	13.74	not statistically significant

Table 11.3. Comparison of Subjects by Experience (AMUR).

11.2 Effects of Spatio-Temporal Independence

As discussed in Section 6.1 Relationship of Space and Time, the goal in tower defense is temporal – maximize the amount of time the towers are active (this is a slight simplification; see Sections 8.4.2 To Convert Wounds to Kills and 8.4.3 To Reduce Overage for details). Despite this, 70% of subjects in the domain novice group and 40% in the domain experienced group attempted to maximize the amount of space a tower covered. They converted a temporal problem into a spatial one, implicitly assuming that the time and space were equivalent, or at least strongly correlated. We wanted to know if this was true.

Subjects were divided into two groups, those who used the spatial ADVANCED MAXIMUM USABLE RANGE (AMUR) strategy set and those who used the temporal DIFFERENTIAL SLOWING (DS) strategy set (for more information, see Section 10.2 Measured Strategies). We asked two questions:

- Does using space as a proxy for time work?
- If so, does treating space and time as independent work better?

11.2.1 Effectiveness of Spatial-Proxying

We compared the performance of the members of the AMUR (spatial) group to three baseline strategies we created. The agents and the strategies they implemented were, in order of increasing sophistication:

- **AI_Random**. Place towers at any legal position.
- **AI_PathAdjacent**. Place towers on any legal position that is immediately next to a path tile.
- **AI_MUR**. The MAXIMUM USABLE RANGE strategy. Place each tower where it covers the most space. This strategy differs from the AMUR strategy only in that this places slowing towers where they cover the most range rather than next to the offense towers.

Each agent attempted each map 100 times. Agents used the exact same towers as the human subjects.

How did the AMUR users do? Compared to AI_Random, quite well. A 2 (Human AMUR vs. AI_Random) x 16 (All maps) ANOVA showed:

- a main effect of map, F(9.47, 1032.70) = 118.32, MSE = 20.14, p < .001
- a group x map interaction, F(9.47, 1032.70) = 20.23, MSE = 20.14, p < .001

- a main effect of group, F(1, 109) = 420.27, MSE = 11.60, p < .001

Across all maps, Human AMUR users (M = 13.91) outperformed AI_Random (M = 8.37). The 5.54 point gap represents a 19.8% advantage. Human AMUR significantly outperformed AI_Random on 14 of the 16 maps and underperformed AI_Random on two.

The story is more complicated for AI_PathAdjacent. A 2 (Human AMUR vs. AI_PathAdjacent) x 16 (All maps) ANOVA showed:

- a main effect of map, F(10.19, 1110.59) = 94.03, MSE = 16.75, p < .001
- a group x map interaction, F(10.19, 1110.59) = 5.47, MSE = 16.75, p < .001
- a marginal main effect of group, F(1, 109) = 3.23, MSE = 13.54, p = .075

Human AMUR's (M = 13.91) and AI_PathAdjacent's (M = 13.38) performance was quite similar. Human AMUR had a large advantage on two maps and modest advantage on one. AI_PathAdjacent had a modest but statistically significant advantage on one map and a modest but marginally significant (p = 0.107) advantage on another.

The average score for humans using AMUR and the AI_MUR benchmark agent were similar but the individual map scores were not. A 2 (Human AMUR vs. AI_MUR) x 16 (All maps) ANOVA showed:

- a main effect of map, F(6.93, 755.37) = 343.34, MSE = 9.67, p < .001
- a group x map interaction, F(6.93, 755.37) = 15.67, MSE = 9.67, p < .001
- a main effect of group, F(1, 109) = 11.12, MSE = 4.87, p = .001

Human AMUR's (M = 13.91) and AI_MUR's (M = 14.49) overall performance was quite similar, with the computer agent holding a slight (2%) numerical edge. The average hides substantial differences in individual map scores. The two performed similarly on only four maps. Human AMUR had a large or modest advantage on two maps while AI_MUR held a large or modest advantage on four.

Human AMUR	AI_Random	AI_PathAdjacent	AI_MUR
Mean	8.37	13.38	14.49
AMUR Advantage Points	5.54	0.53	-0.58
AMUR Advantage Percentage	20%	2%	-2%

Table 11.4. Human use of the Advanced Maximum Usable Range (AMUR) strategy compared to baseline synthetic benchmarks. Average score of the humans using the spatial strategy ADVANCED MAXIMUM USABLE RANGE compared to baseline scores.

Humans using the spatial AMUR strategy set substantially outperformed chance, indicating that there is value in spatial proxying. Its performance was similar to the two simple strategies, path adjacency and maximum usable range. We found this slightly surprising. One possibility is that the advantage humans have in using a more advanced strategy are offset by the computer's ability to directly measure usable range rather than approximate it as humans do. Another possibility is that the differences between the MUR and AMUR strategies are less important than we initially believed.

11.2.2 Spatial Proxying vs. Direct Temporal Manipulation

The spatial AMUR strategy set, which uses space as a proxy for time and attempts to maximize space, performs significantly better than chance. How does it fare relative to the temporal DS strategy set, which treats space and time as independent and attempts to directly maximize time?

A 2 (Human AMUR vs. Human DS) x 16 (All maps) mixed factor ANOVA showed:

- a main effect of map, F(15, 210) = 58.20, MSE = 7.39, p < .001
- a group x map interaction, F(15, 210) = 1.91, MSE = 7.39, p = .024
- a main effect of group, F(1, 14) = 12.62, MSE = 20.55, p = .003

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human DS	**Human AMUR**			
Finger	18.60 (2.49)	19.41 (1.15)	-0.81	Essentially no difference	Nonsig.
NoSupport	16.35 (1.44)	17.14 (3.80)	-0.79	Essentially no difference	Nonsig.
Switchback	22.85 (3.97)	22.45 (2.65)	0.40	Essentially no difference	Nonsig.
DoTheSplits	6.25 (2.05)	5.70 (0.89)	0.55	Essentially no difference	Nonsig.
UpAndDown	17.60 (2.71)	16.98 (5.21)	0.62	Essentially no difference	Nonsig.
Ladder	11.85 (1.07)	10.81 (1.10)	1.04	Slight Human DiffSlow Advantage	p = .101
RoundTheTwist	26.35 (1.75)	24.89 (2.77)	1.46	Slight Human DiffSlow Advantage	Nonsig.
SlimPickings	18.95 (4.42)	17.32 (3.47)	1.63	Slight Human DiffSlow Advantage	Nonsig.
2in1out	9.40 (1.72)	7.48 (0.87)	1.92	Slight Human DiffSlow Advantage	p = .009**
TheFrog	14.35 (2.00)	12.14 (2.67)	2.21	Modest Human DiffSlow Advantage	p = .122
Bottleneck	11.65 (1.08)	8.09 (1.06)	3.56	Modest Human DiffSlow Advantage	p < .001***
NoLeftTurns	9.75 (4.60)	5.89 (1.33)	3.86	Modest Human DiffSlow Advantage	p = .019*
SnakingPath	21.00 (6.20)	17.02 (1.63)	3.98	Modest Human DiffSlow Advantage	p = .059
Elemental_ish	20.30 (5.46)	15.89 (4.13)	4.41	Large Human DiffSlow Advantage	p = .094
Void	18.45 (2.43)	13.51 (3.19)	4.94	Large Human DiffSlow Advantage	p = .008**
Pathways	13.60 (2.92)	7.84 (1.80)	5.76	Large Human DiffSlow Advantage	p < .001***

Table 11.5. Average score per map for humans using the Advanced Maximum Usable Range (AMUR) strategy compared to humans using a Differential Slowing (DS) strategy (Advanced Difficulty). Sorted by magnitude of human DS advantage. * indicates p < .05, ** p < .01, *** p < .001.

The temporal DS strategy set (M = 16.08) outperformed the spatial AMUR strategy set (M = 13.91) across all 16 maps. The 2.17 point difference represents an 8% advantage for the temporal DS strategy set. This number hides an important point – DS does substantially better on some maps and no better on others. This fact is reflected in the modest but significant map x group interaction. The DS advantage for the individual maps is shown in Table 11.5.

Does treating time and space as independent work better than using space as a proxy for time? The answer is, it depends on the map. For 44% of the maps, using the temporal DS strategy set failed to show a statistically significant advantage. DS has a slight advantage on two maps, a modest advantage on four and a large advantage on two.

Why does the map matter? There are several reasons. Some maps are so simple that any strategy works well. On other maps, the DS strategy is either difficult or impossible to use (for a detailed explanation, see 8.2.3 To combat attack window density). In contrast, some maps cannot be solved without using a DS strategy. A full discussion of this issue is given in Appendix D Detailed Statistical Analysis of Experiment 2 (specifically Section D.1.2 Map Categories).

Mean		DS
Human AMUR	**Human DS**	**Advantage**
13.91	16.08	7.8%

Table 11.6. Comparison of the two human strategies, the spatial strategy Advanced Maximum Usable Range (AMUR) and the temporal strategy Differential Slowing (DS). [further table note deleted here/incorrect/misleading text]

11.3 Human vs. Computer

We developed two computer agents, AI_AMUR and AI_DS, that play the same strategies as the humans in Experiment 2. We refer to these as *strategy agents*, to differentiate them from the baseline agents.

11.3.1 Strategy AIs vs. Baseline AIs

How did the strategy agents do relative to the baseline strategies? Quite well. Using a slightly different test (see Appendix D Detailed Statistical Analysis of Experiment 2 for details), AI_AMUR (M = 21.16) and AI_DS (M = 23.91) scored higher than AI_Random (M = 9.19), AI_PathAdjacent (M = 19.36) and AI_MUR (M = 19.67). This is slightly better than human AMUR users did relative to these benchmarks. As with human AMUR users, the strategy agents outperformed the baseline agents more on some maps than others.

	AI_Random	AI_PathAdjacent	AI_MUR	AI_AMUR	AI_DS
AI_AMUR	42.8%	6.4%	5.3%	-	-9.8%
AI_DS	52.6%	16.3%	15.1%	9.8%	-

Table 11.7. Strategy AIs compared to Baseline AIs. Values represent the advantage the agent on the side has over the agent on top.

11.3.2 AIs vs. Domain Experience

Returning to the issue of domain experience, how do subjects at different levels of experience do relative to the strategy AIs? The answer is unclear. The domain novice group (M = 20.56) underperformed both strategy agents. The domain experienced group (M = 22.56) outperformed strategy agent AI_AMUR (M = 21.16) but not AI_DS (M = 23.91).

	AI_Random	AI_PathAdjacent	AI_MUR	AI_AMUR	AI_DS
Novice	40.6%	4.3%	3.2%	-2.1%	-12.0%
Experienced	47.8%	11.4%	10.3%	5.0%	-4.8%

Table 11.8. Strategy AIs compared to humans at different experience levels. Values represent the advantage the humans at a given experience level (rows) have over the computer agents (columns).

Earlier we showed that experienced subjects only perform modestly better than domain novices but experience substantially affects which strategies a subject uses and which strategies a subject uses substantially affects their scores. AI_DS outperforms experienced humans but not to the same degree it outperforms domain novices. Is this because experienced subjects are moderately and uniformly better than domain novices but worse than AI_DS or are the gains due predominantly to the experienced subjects who use different strategies? We look at the effect of strategy in the next section.

11.3.3 Strategy AIs vs. Human Strategies

How well do the strategy agents perform relative to humans using the same strategy? The answer is, about the same.

In the case of the spatial AMUR strategy set, a 2 (group: AI_AMUR vs. experienced AMUR) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(7.63, 778.01) = 100.87, MSE = 2.42, p < .001
- a group x map interaction, F(7.63, 778.01) = 35.37, MSE = 2.42, p < .001
- no effect of group, F(1, 102) = 1.64, MSE = 1.90, p =.20

Across non-ceiling maps (see Appendix D Detailed Statistical Analysis of Experiment 2 for details) there was no significant difference between AI_AMUR (21.16) and human AMUR users with domain experience (21.43).

In the case of the temporal DS strategy set, a 2 (group: AI_DS vs. human DS) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(6.16, 633.98) = 115.05, MSE = 1.52, p < .001
- a group x map interaction, F(6.16, 633.98) = 39.45, MSE = 1.52, p < .001
- no main effect of group, F(1, 103) = 2.40, MSE = 0.91, p = .125

Across all non-ceiling maps, there was no significant difference between AI_DS (M = 23.91) and human DS users (M = 23.71). Interestingly, although playing the same strategy and earning the same average score, scores varied significantly on 73% of the maps, at times by substantial amounts.

It is unclear why the individual map scores for two groups using the same strategy should vary so much. The human group has significantly larger standard deviations but these do not appear to be correlated with performance advantage. Advantage also does not appear to correlate with map difficulty, as measured by mean scores. Amenability to DS strategies is also not correlated – of the five maps strongly amenable to DS strategies, humans scored better on two, AI_DS scored better on two and they did the same on the last.

We have succeeded in creating an agent that performs at the same level as a human with domain experience using an advanced strategy but the variance in individual map scores suggests that the agent can be further improved.

	AI_AMUR	**AI_DS**
Novice AMUR	-2.5%	-12.3%
Experienced AMUR	non-significant	-8.9%
DS	9.1%	non-significant

Table 11.9. Humans vs. computer agents, both within and across strategies. Values represent the advantage the human strategy (rows) has over the computer agent (columns).

Map	Mean (SD)		Diff.	Comparison	Significance
	Human DS	AI_DS			
DoTheSplits	17.20 (4.44)	24.05 (0.80)	-6.85	Large AI_DS advantage	p < .001***
Bottleneck	22.40 (2.07)	24.68 (1.01)	-2.28	Modest AI_DS advantage	p < .001***
Void	25.60 (3.05)	26.84 (0.56)	-1.24	Slight AI_DS advantage	p = .001**
2in1out	20.80 (1.92)	21.81 (0.62)	-1.01	Slight AI_DS advantage	p = .003**
UpAndDown	27.40 (0.89)	27.54 (0.70)	-0.14	Essentially no difference	Nonsig.
Finger	26.00 (2.12)	26.00 (0.00)	0.00	Essentially no difference	Nonsig.
NoLeftTurns	20.40 (6.66)	20.40 (0.49)	0.00	Essentially no difference	Nonsig.
Ladder	23.80 (1.92)	22.11 (1.10)	1.69	Slight Human DS advantage	p = .002**
Pathways	26.40 (2.61)	24.34 (1.22)	2.06	Modest Human DS advantage	p = .001**
TheFrog	25.80 (1.79)	23.66 (0.57)	2.14	Modest Human DS advantage	p < .001***
NoSupport	25.00 (2.55)	21.62 (0.49)	3.38	Modest Human DS advantage	p < .001***

Table 11.10. Average score per map for humans using the Differential Slowing (DS) strategy compared to a computer agent using the same strategy. Sorted by magnitude of human DS advantage. * indicates p < .05, ** p < .01, *** p < .001.

11.4 Conclusions

We began by asking five questions about three topics: domain experience, space-time conflation and humans vs. computers.

11.4.1 Domain Experience

The task we used was new to all of our subjects but half our subjects had experience with a task that was modestly similar. Humans are excellent at transferring what they've learned solving one type of problem to help solve new problems, at least if the tasks look fairly similar. Half our subjects had placed different types of towers on different maps to stop different types of creeps. We asked two questions. First, does this experience help? The short answer is, surprisingly, not that much. Subjects with experience in similar domains scored 7% higher on the 11 maps that had discriminatory power. Some of that might be explained by the fact that we had constrained the task a great deal, forcing subjects to use a set of towers we chose and, by extension, limiting the number of strategies they could use, potentially ruling out strategies they used in other domains. Numerous subjects complained that GopherTD did not have an area of effect (see glossary) tower, which are the cornerstone of their strategies in other tower defense games. Furthermore, some subjects had significantly less experience than others, and we noted that the number of different tower defense games they'd tried had an impact on performance, with those who had only played a single tower defense game, no matter how often they had played it, abstracting less and consequently carrying over flawed assumptions.

Even so, we were surprised to find that experience didn't count for more.

While experience in and of itself did not seem to have a large effect on performance, it had a tremendous impact on strategy selection. The second question we asked is whether subjects' experience with a similar domain would decrease the space-time conflation we'd observed in earlier experiments. The answer is a resounding yes. Half of the subjects with domain experience used the DS strategy, which attempted to directly maximize tower activation time. None of the domain novices used a temporal strategy. Given that the temporal DS strategy significantly outperformed the spatial AMUR strategy, this is an important finding. Given that strategies exploit affordances and the affordances are a significant part of the problem representation, this implies that experience, where it leads to strategy changes, causes subjects to form more sophisticated representations of a domain.

So what can we say about experience? If those with experience continue to use the strategies that they use as novices, experience leads to them doing slightly better, implying that experience has helped them improve their execution of their strategies. The big advantage of experience, however, is that it teaches you new and more sophisticated ways of representing the problem, leading to new and more effective strategies, and it is those new strategies, more than experience directly, that has a significant impact on performance.

11.4.2 Space-Time Conflation

One way to solve a problem involving time is to act like it's a problem about space. Rather than maximize time, maximize space and, if space and time are correlated, maximizing one will maximize the other.

We asked two questions. The first was, how realistic is it to use space as a proxy for time? Although counter-intuitive, treating time as space is both a common phenomena, both in this domain and others, and appears to work well. The average of those using the spatial-proxying strategy ADVANCED MAXIMUM USABLE RANGE was 42 percentage points above that of a random baseline, outperforming chance on every map.

The second question was, does attempting to directly maximize time lead to better performance than using space as a proxy? The answer is yes, but not nearly to the extent that spatial-proxying outperforms chance. The temporal DIFFERENTIAL SLOWING strategy outperformed the spatial-proxying AMUR strategy by an average of 10 percentage points. That gain depended on the map, ranging from a low of 0 to a high of 34 percentage points.

What can we say about space-time conflation? When the goal is temporal, maximizing time directly is more accurate but converting the problem to a spatial problem and solving that works remarkably well. If solving a spatial problem is significantly easier, one can understand why someone might be willing to accept the slight loss of accuracy and use space as a proxy for time.

11.4.3 Human vs. Computer

We argued earlier that one cannot honestly claim that a paper model is accurate until it has been implemented in a computer. Implementing the human strategies in a computer forced us to face several subtle issues we had not noticed in our paper model. In order for us to claim that we did so successfully requires us to compare the performance of the computer against the humans.

We also noted earlier that it would not be an entirely fair comparison. Computers are better at measuring quantifiable properties while humans, at least compared to the current AI, think more flexibly, allowing them to address problem-specific subtleties. We wanted to see whether the computer or human advantage would outweigh the other.

What did we learn? On average, the humans and AIs performed the same. There were a few maps where either the computer or humans did better but these averaged out, resulting in no statistically significant difference between the groups.

A qualitative review of the solutions to those maps where statistically significant differences were found suggests that the AI is generating the same type of solution as humans. Where the AI underperformed, it was due to the AI lacking sophistication in a one of the non-primary strategies, typically in how ties are broken between positions with equal utility. Where the humans underperformed, it was typically because one of the subjects made an error. Notes from subject interviews show that these differences occurred on maps where subjects stated they weren't thinking and "did something dumb". AI errors were due to inadequate supporting strategies while human errors were due to execution, specifically consistency and perception (i.e., not noticing a property of the map). At this point we can say that both the computers and humans have weaknesses unique to each group but we cannot claim that either has an overall advantage over the other.

Chapter 12

Conclusions

12.1 Summary

In this work we've spent a lot of time trying to determine how, precisely, the human problem solving process works when dealing with spatio-temporal problems.

12.1.1 Problem Solving

While we have looked at a single spatio-temporal task (spatio-temporal reasoning being a critical part of functioning in the real world), we believe that the lessons learned apply to the larger world of human capabilities in solving real-world problems. These lessons include the following.

Elements of Problem Solving: Affordances and Strategies, not Facts and Algorithms. Much of the existing work in spatial representation has focused on finding a general purpose method of describing positions and spatial relationships. Much of the work on problem solving involves developing efficient ways of executing large numbers of rules or searching in large state spaces in an effort to find meaning in these facts. This is not what we found in human problem solvers. When solving problems, both representation and reasoning are specific to the problem being solved. Our subjects did not represent the space, they represented the actions and outcomes afforded by the space. Their reasoning did not use a general purpose reasoner, spending relatively large amounts of processing time and memory to draw inferences and produce new, intermediate facts, they extended their affordance-based representations to the obvious actions that were afforded. Complex problems were solved not by complex reasoning but by fast, simple, straight-forward responses to affordance-based representations.

Data as Actions: Reasoning by Representation. Problems are represented not as a set of features but a set of affordances. Subjects were often unable to recall specific spatial configurations. When asked to describe a map, they would not list spatial features such as "there were six corners and a long path down the side", they would list goal-relevant affordances such as "that was the map with the bottleneck in the middle" and "it was the map where the two lines reached the end of the map at different times". When a problem has been parsed into a set of affordances such as "red towers will cover the most area if placed here" and "the creeps are separated after this corner", the reasoning step is almost trivial. As a result, the most important step in solving a problem might not be thinking about the facts but perceiving the affordances.

Problem Solving: Sub-Goals, Strategy Sets and Compositional Reasoning. How do human problem solvers solve complex problems? From our research the answer appears to be, they don't. They convert complex problems into sets of simple ones. They do not solve these problems with a complex, general purpose algorithm but by a set of simple, focused strategies. Individually, the strategies are often unimpressive. The power of human reasoning comes not from any single strategy but from combining many strategies. We are not the first to note that humans use "fast and frugal heuristics". Many studies have shown subjects using a single, simple strategy to answer a "small" question (judging which of two cities is larger, comparing fractions, etc.), have a set of simple strategies and select the strategy to use based on the affordances of the problem but we believe we are the first to show that complex problems are solved compositionally, combining multiple strategies from multiple categories of strategies to solve multiple goals, with part of the intelligence lying outside the individual strategies and in the meta-process of selecting and combining them.

Tightly Coupled Goal-Representation-Action. While we saw many affordances and representations that could be used in other contexts, at no point did we witness someone using a general purpose representation or algorithm. Everything was specific to the problem. For example, subjects had the goal of getting the maximum score on a problem. When they did not achieve the maximum score, they attributed this to their towers not having enough time to attack. At this point they formed a new goal, place towers where they cover the most area, which they determine is a U-turn, leading to the less abstract and easier to process goal, place towers around U-turns. They create and test the new solution and notice that a creep spends significantly less time in the range of a tower that is inside a corner. They realize this is because towers inside corners cover half the room of towers along walls. They form the new strategy, avoid interior corners. On their next attempt, they notice where the interior corners are and place the towers around the U-turn while avoiding the interior corners. Through this process of goal formation, perception, representation and strategy, subjects build up their problem solving approach. The strategy of avoiding interior corners is closely tied to the representation of the affordance *interior corner*, that representation, in turn, is tied to the perception of wasted space and that perception is tied to the goal of maximizing space.

Avoid Powerful, General-Purpose Algorithms. In Related Work we mentioned studies of math students that showed that even when students were taught to use a single method of solving a given class of math problems, students used self-taught strategies that required little thought or memory but only worked on a small subset of problems. While our subjects solved complex problems using multiple strategies concurrently, the nature of the strategies was similar. Strategies such as place towers after corners and place slowing towers before offense towers rather than after were simple, requiring trivial amounts of processing and working memory. Most of the effort went into recognizing opportunities afforded by the problem. Contrary to what many scholars suggest, we found that the key to flexible thinking was not abstraction and generalization, it was using perception to quickly create concrete, goal-specific representations and strategies.

Representational Transference: Solving Novel Problems by Matching Affordances, Not Structure or Features. Much of the literature has suggested that the human ability to leverage past experience to solve novel problems is due to analogical reasoning and the ability to say that two problems are similar in structure or features. We saw no evidence of this in our studies. Upon encountering new problems, subjects did not characterize new problems as U-turn problems (feature) or looping problems (structure), they described them as "problems I can use strategy X on". Problems were matched on affordances, on what could be done in the problem rather than any feature specific to the problem itself. Unlike previous work suggesting analogical reasoning, in which a solution to one problem is used on a new one, subjects did not try to borrow and adapt a solution from a single base case. They looked for affordances and used the corresponding strategies. These strategies came from multiple problems and the set of strategies used on the novel problem was potentially a novel combination.

Substantial Variation in Problem Solving Style. When we began this work, based on our own experience with the domain, we expected to find three strategies. Instead, across all maps, 58 subjects used 74 distinct strategies. No two subjects solved the same problem the same way. We believe that we are the first to show that this level of variation exists in problem solving. We attribute this variance to the task complexity. The task is more complex and allows more solutions than most laboratory tasks but is simple enough to allow precise measurement.

Individual Variance is Compositional Variance. We believe that we are the first to show that, while each subject was essentially unique in how they solved problems, this variance was not due to each person having a different strategy. Many strategies were used by multiple subjects and a few were used by the majority. These strategies, however, were used concurrently and in combination with several other strategies. Differences in problem solving style were due not to any individual strategy but to the set of strategies used, with each subject using a unique combination of strategies.

12.1.2 Spatio-Temporal Reasoning

We uncovered a few lessons specific to spatio-temporal problem solving.

Spatial Reasoning without Spatial Representation. In our studies, spatial reasoning was done, counter-intuitively, without spatial representation. In tasks where the goal is to represent space (e.g., object recognition, sketch understanding, navigation), spatial representation is unavoidable. As these are the most common spatial tasks studied in artificial intelligence, a lot of work has gone into spatial representation. In problem solving, though, solutions might take place in space but the problem solver doesn't need a description of the space, they need to know what they can achieve in the space. In this work, subjects examined the map to determine what the affordances were, but once they had found the affordances they appeared to pay no attention to the structure of the map or the parts of the map that were not part of a strategy.

Spatio-Temporal Conflation and Spatial Proxying. A surprising finding in this study was that, despite the goal of the task being to maximize a temporal property, most subjects implicitly converted the problem into a spatial one and solved that. We were also surprised to find that this approach worked surprisingly well.

Space, Time and Multiple Objects: Buffering and Attack Window Decay. When a single object is moving at a constant rate through space, space and time are perfectly correlated and there is no penalty in converting a temporal problem into a spatial one. This correlation breaks when there are multiple objects and a processing entity (e.g., a tower) can only process one at a time. When both objects can be processed, one captures the processing object's attention (i.e., the tower attacks it) while the other moves through space untouched. When the processing object is done processing the first object, it has less time to process the second one. We refer to this as buffering and attack window decay and it compresses time without effecting space. In a spatio-temporal environment with multiple objects, optimal performance depends on explicitly manipulating temporal windows.

12.1.3 Data Collection

To better understand spatio-temporal reasoning in general and the tower defense domain specifically, we collected and documented a substantial amount of information.

Local Position Representation: Target, Relationship and Anchor. The placement decision for a tower can be specified in terms of the item to be placed (the placement target), an item it is placed relative to (the placement anchor) and a relationship between the two. The number of items we observed in each category was surprisingly large – seven placement targets, 19 placement relationships and 36 placement anchors.

Solution Representation. We identified several properties necessary to properly describe a generated solution. In addition to placement decisions, towers have roles, belong to functional groups, follow a distribution and are placed in consistent placement order. These properties not only make it possible to describe a solution, they underlie many of the strategies used by subjects.

Strategy Catalog. We captured details and motivations for 74 strategies in seven categories. This is by no means a complete list. We only cataloged those strategies that we observed subjects using and those subjects had no previous experience with the task. While we believe continued study, especially of people with GopherTD experience, would turn up many more strategies, we believe the ones we captured give a good idea of the types of strategies available.

12.1.4 Computer Models

Finally, in the service of building computer agents better able to deal with the real world, we developed the following:

GopherTD and Game-Based Investigation Tools. To study decision making in domains with significant spatio-temporal components, we developed GopherTD. Based on the popular commercial tower defense game Vector TD, GopherTD supported a number of research needs. GopherTD controlled map presentation in testing, logged player actions and time spent on them, could view and re-run logged solutions, supported AI agents, supported batch runs and contained a number of visualization tools. The Strategy Explorer allowed the experimenter to create strategies and solve maps by applying a set of such strategies.

Strategy-Based Reasoning Framework. The agents we created use the strategy-based reasoning framework we developed. Agents describe their placement strategies in terms of placement decisions (e.g., place slowing tower's range so that it overlaps the start of an offense tower's range) and delegate instantiation to a general spatio-temporal solver. Agents are primarily declarative, describing their set of strategies in terms of affordances. As such, most agents are less than 20 lines of code.

Filtering as Strategy Combination. In order to use a set of strategies, the strategies must somehow be combined. In our work, we chose to model strategies as filters. Combining strategies was accomplished by passing a set of candidate positions through a set of filters. Strategies such as "place the range so that it overlaps an offensive tower's range" and "do not place towers on interior corners" were combined by using each strategy to remove candidates that did not match the strategy's criteria. Order of filtration was irrelevant beyond the constraint that ordinal scoring filters must go last (e.g., select position with the most usable range). This method of combining strategies was simple to implement and debug and relatively fast.

Spatial Affordance Query System. We have argued throughout this work that the key to human reasoning is focusing on the opportunities afforded by the environment. We have also pointed out that much of the intelligence in spatio-temporal reasoning is being able to perceive these affordances. Correspondingly, the majority of code and effort in this system is in SAQS, the Spatial Affordance Query System. An annotated map model is created from the base map geometry, generating meta-data to support affordance queries. When an agent passes its set of strategies to the solver, the solver instantiates these strategies by making a series of calls to SAQS to find the matching affordances. A significant part of the work in this research went into defining and identifying spatio-temporal affordances and the creation of the spatio-temporal affordance query system.

Quantifying Time from Space: The al Measure. The majority of subjects treated time and space as equivalent, and in many instances the two properties could be profitably thought of as correlated. There is a relationship between time and space but it is not as straight forward as many subjects treated it. Subjects who understood this performed better than those who did not but still had problems as they found quantifying aggregate time over a set of agents more difficult than quantifying space. The al (agent length) measure unifies time and space, describing the size of an attack window in terms that can be directly converted to time or space. This allows the problem solver to use space as an accurate measure of time and vice-versa.

Spatio-Temporal Agents. We developed a number of spatio-temporal reasoning agents. Some modeled specific subjects, others modeled specific strategies. The strategy agents performed at roughly the same level as humans, which was one of the primary goals of this research.

Tower Defense Domain. While we most certainly did not create the idea of tower defense puzzles (they grew organically from the collective experience of tens of millions of real-time strategy game players), we wish to argue that the domain, which to the best of our knowledge has never been studied either in computer science, cognitive science or experimental psychology, is an excellent domain to study decision making, problem solving and spatio-temporal reasoning. Like blocks worlds, sliding tile puzzles, cryptarithmetic problems and Raven's Progressive Matrices, tower defense is small and simple enough to construct quantifiable, repeatable, controlled tests that abstract out unwanted variables. Unlike those, it is complex enough to study behavior at the level of real-world tasks, increasing the probability that the lessons learned will scale to issues of interest.

12.2 Open Research Areas

The work described in this thesis covers issues in psychology and computer science, problem solving and spatio-temporal reasoning, complex reasoning and bounded rationality, strategy cataloging and data representation. We used a new domain to study a new task in a new class of problems. While we feel we have learned a lot, it is expected that such research generates more questions than answers. Below are some of the many issues that we began investigating but eventually had to postpone due to resource limitations.

12.2.1 Limitations of Current Work

Several years of work went into the research reported here but several more would be needed to accomplish everything we would have liked. Among the things we were unable to get to are:

Advanced Strategies and Representations. We collected strategies and representations from subjects who either had no experience with tower defense puzzles or who had experience with different types of tower defense puzzles but not the type we used. Our initial reasoning for excluding subjects with GopherTD/Vector TD experience is that we were studying transfer learning and thus needed people for whom these puzzles would be new. As a result, the strategies we collected are those of people who have spent no more than one hour of training in this domain. Had we studied more experienced subjects, we believe we would have discovered more sophisticated strategies that delved more deeply into intricacies of spatio-temporal reasoning.

Breadth of Implemented Strategies. We created seven agents. Three implemented baseline strategies, two implemented the strategies ADVANCED MAXIMUM USABLE RANGE and DIFFERENTIAL SLOWING and two mimicked the behavior of subjects N0100 and E0100. We implemented those strategies needed for these agents plus many more to support the Strategy Explorer, which allowed users to create strategy sets on the fly, but we did not implement all of the strategies we had collected. In particular, we did not implement Strategy TP5: TEMPORAL ATTACK WINDOW SEPARATION, an advanced temporal strategy. To be more accurate, we never successfully implemented it; all temporal strategies were difficult to implement but attack window separation was particularly difficult.

Unit Selection Strategies. We have argued that subjects select strategies based on affordances. Towers constitute part of those affordances. In our performance studies (Experiment 2), in order to be able to isolate the performance of placement strategies, all subjects were required to use the same set of towers. To be able to compare these results to the performance of our agents, our agents were also instructed to use a specific set of towers. We did not implement any tower selection strategies.

In Experiment 2, subjects used offense towers that were low powered, had a short range but were extremely fast. This ruled out the use of strategies that required powerful towers or towers with significantly larger ranges. Presumably, some maps are more amenable to strategies that use fast towers and others are more amenable to strategies that require towers with large ranges. For this reason, tower selection, or more generally tool selection, is an important part of problem solving.

Strategy Selection Strategies. It was our belief that the towers and placement strategies one uses depends upon the specifics of the map being solved. We based this on an analysis of two domain experts. In our studies, however, subjects used the same towers and strategies on all maps. We believe this is because these subjects only had an hour of experience with GopherTD and that this amount of experience is insufficient to get subjects to an expert's level of sophistication. We were unable to gather sufficient data on strategy selection strategies and as a result chose not to create an agent with this capability.

Conflict Resolution. The primary agents we implemented were designed to play a specific strategy. This allowed us to test the performance of the strategy independent of other concerns. As such, we had no test bed for studying strategy selection and as a result had no test bed for studying strategy combination beyond those strategies explicitly assigned to the strategy agents.

12.2.2 The Instantiation Problem

Much has been written about how concrete knowledge can or should be abstracted into general principles but significantly less has been written about how to convert these abstractions and general principles back into concrete solutions. In the work described here, subjects used compositional reasoning – rather than use a single, comprehensive strategy, they used a set of focused strategies drawn from a larger set of known strategies and combined them to form a solution. We know of no work beyond our own that has investigated how this combination process works.

Selection. Part of this process involves selecting an arbitrarily sized set of strategies to apply to a given problem. These strategies are chosen based on the affordances of the problem but a meta-process of making sure strategies are symbiotic or at least not in conflict must occur.

Conflict Resolution. As the strategy catalog shows, many strategies have no overlap and can be executed in parallel. If two strategies are mutually exclusive or potentially in conflict, these conflicts must be arbitrated, potentially by ordering or finding a compromise between them. In most cases there will be resource conflicts – given a fixed number of towers or resources to obtain towers, the agent must decide how many towers are allocated to each strategy.

Ordering. Many strategies have no dependence on other strategies but strategies such as "place near an offense tower" and "surround the kill zone" use placement anchors that must exist before the strategy can be instantiated.

Concurrent Execution. We implemented strategies at the tower level. This works in the majority of instances we modeled but would not have worked for strategies that involve building composite structures. For example, some subjects created staggered tower chains on each side of a path, forming a structure that looks like teeth in zipper. We created a special object (`ConcretePattern`) for placing objects according to a fixed size structure but this issue needs further investigation.

Modifiers. Some strategies modify how other strategies work. For example, the strategy "don't place towers on interior corners" does not place towers itself but affects how all other strategies place towers. We implemented strategies as a set of filters, which worked well for the strategies we implemented but we have not confirmed that this approach will work for every type of modifier strategy.

Iteration. Our agents determined all placements in a single pass through the strategies. This did not match how humans placed their towers. Humans created candidate solutions, studied them and then modified them. It was common for subjects to remove towers they had placed earlier, and in some instances subjects removed all towers from an area and moved to a new one. We did not use an iterative approach because, due to computer advantages in measurement and affordance recognition, we did not feel that we needed to, but it is possible that the iterative approach used by humans has value beyond being an aid in perception.

Optimality. Humans appeared to have little problem combining strategies but we have no way of gauging the quality of these combinations. An agent would want to find the optimal combination. How this can be accomplished is an open issue.

We have documented some of the methods subjects used to combine strategies and we hope we have shown the importance of this topic. This topic deserves more study.

12.2.3 Problem Characterization

In Chapter 9 – Performance we grouped five maps together with the justification that they were unusually amenable to DIFFERENTIAL SLOWING strategies. This was a subjective judgment – we did not justify this grouping by pointing to an objective property these maps shared. The reason for this is that we were unable to find one. Two domain experts individually selected these maps with perfect inter-rater agreement, strategy use data supports this grouping and the choice seems like common sense to those who we’ve shown it to but a structural analysis of 24 properties of the maps found no property or combination of properties that selected these specific maps. As humans, we know these maps are the same but we can't seem to articulate why or find the reason in the features we've captured. There remains a gap between how humans view problems and how we've computationally represented them.

12.2.4 Expertise

Some subjects consistently perform better than others. In our research we found that much of this can be explained by choice of strategy but even within strategy some subjects consistently outperform others.

There is also the issue of strategy selection – why did some subjects use a superior strategy and others did not. We believe this is due in part to perception – subjects who used a given strategy almost always justified it by pointing to something they noticed in a previous test – but why do some subjects notice these issues and others do not? Only subjects with domain experience used advanced strategies. If perception is the reason that one group uses an advanced strategy and another does not, this implies that experience has an impact on perception. Some subjects with experience continued to use novice strategies. It is unknown why experience benefits one person more than another.

12.2.5 Feature Learning

Our Spatial Affordance Query System supports a large number of important and occasionally complex affordances, but all of these were manually authored. Humans have the ability to learn new properties which they can use for reasoning. We have not yet determined how, precisely, this happens or how we can implement this capability in computers.

Based on what we observed in our studies, we believe the answer lies in perception. We did not initially expect perception to be an important part of representation or learning as it eventually became clear it was. Perhaps the biggest differentiator between high and low scoring subjects was that high scoring subjects (who were invariably subjects with domain experience) were more likely to notice small things that had gone wrong in their solution. Once these problems were noticed, new goals, representations and strategies quickly followed. Perceived problems were generally goal-driven – the creeps made it past the primary towers and there were no later towers to catch them, the creeps moved too fast in this area, the freezing towers all tried to freeze the creep at the front of the line and so most of them wasted their shot, etc. Relatively little work was required to convert these observations to affordance-based representations and a related strategy.

Without goal-driven perception, it seems unlikely that problem solving would work, yet we know of relatively little work in artificial intelligence that has examined this issue (while feedback systems such as back propagation adjust a system, they do not create new representations or strategies). Quite a bit of work has been done on this in psychology but not much of it has been done in the context of complex problem solving. We believe that the role of perception in the development of new representations and strategies deserves further study.

12.2.6 Strategy Learning

How do subjects learn new strategies? We have suggested that strategies are not a matter of insight or genius so much as they are reactions to affordances which themselves are reactions to goals, which in turn are reactions to observations. While the initial evidence suggests that this is at least one way in which strategies are learned, we have not conducted a thorough study to show that this is the only way. The precise mechanisms still need to be studied and methods of implementing this capability in computers need to be developed.

12.2.7 Optimization

Strategies appear to allow subjects to generate quality solutions to novel problems the first time those problems are encountered. There appears to be a limit, however, on how close a strategy can get to an optimal answer. As discussed in Appendix B – Perfect Scores, strategies often placed towers in approximately correct position but finding the optimal solution sometimes required ***jitter***; pieces had to be moved one or two tiles at random before an optimal solution was found. While there was uncertainly a reason why the new position was better than the one near it, it was not found in the initial retrospective analysis. The solution worked, we just didn't know why.

In Experiment 1 we gave subjects five attempts to solve certain maps. As in our search for perfect solutions, subjects used strategies to get pieces in roughly the right place but used a different process to get them to their precise (?) location. More study on optimizing the results of strategy application is needed.

12.2.8 Learning from Experience

People get better with experience, but why? We have noted that not everyone benefits from experience to the same degree, that experience likely has an impact on strategy use and have suggested that perception plays a key role in learning but is that the full story?

In Experiment 1, subjects were given multiple attempts to solve certain maps. We noticed a consistent pattern of behavior. Subjects created and tested an initial solution. When they failed to get a perfect score, they attributed the error to one part of the solution. On the first failure, this was always the choice of towers. If subjects used green towers, they switched to red and vice versa. On the second failure, subjects consistently blamed the map location, switched back to their original set of towers and moved them to a distant part of the map. Both were large changes. On subsequent failures, the explanations and changes were smaller, with subjects eventually narrowing changes to moving a single tower a single space.

We have no idea why towers were consistently blamed before positions. Perhaps subjects preferred to blame the tools rather than the person using them? Whatever the situation, there remains work to be done on how people benefit from experience and how computers could do the same.

12.2.9 Scalability

We have argued that strategies allow humans to quickly deal with a large, noisy, ever changing world. We believe strategies scale better than brute force. Evidence from artificial intelligence, where computers are perfect at checkers, strong in chess and weak in go, suggests that as the search space grows, brute force AI techniques fall behind human abilities. We tested GopherTD puzzles on a 22x18 grid. We are confident that both humans and our strategy agents would perform just as well on a grid an order of magnitude larger but this hypothesis has not yet been tested.

12.2.10 Imperfect Information, Dynamic Environments and Real-Time Reactivity

We have argued that an advantage of human decision making is that simple, focused strategies allow the decision maker to react quickly while strategy sets allow the decision maker to adapt their approach to the affordances of a specific problem. We tested decision making in GopherTD, which gives perfect information and, other than the movement of the creeps, a static environment. A logical next step is to see how strategy-based agents perform in imperfect information, dynamic environments. We have already started thinking about how we could contribute to the domain of military operation in urban terrain (MOUT) simulations.

It is said that plans rarely survive contact with reality. A plan to secure a building involves breaking through a glass window but the window turns out to be unbreakable plexiglass. The route to an extraction zone is supposed to have only two guards but there are five. A city square is supposed to be empty at night but on the night the plan is executed the square is full of civilians. It is common for soldiers in the field to encounter situations where they must rapidly adjust not only their plan but their tactical goals and strategies. To do this, they must quickly recognize changes in the environment, determine whether these changes invalidate their current plan, identify the new set of threats and opportunities, determine which strategies are relevant in the new situation and form a new plan.

How might we use strategy-based and spatio-temporal reasoning in the MOUT domain? Consider the domain of securing a building. A set of agents must enter from one or more entrances to suppress an enemy within. To be maximally effective, agents must coordinate their movements to be at the right places at the right times. Because the interior of the building might not be fully known, it is important to be able to quickly react to new information. In this domain we might wish to create a plan to secure the building, evaluate a plan given to us, train soldiers via an intelligent tutoring or after-action review system, design simulated agents (friendly or opposing force) to populate a scenario or build full simulations.

For a concrete example, consider the situation in which a squad of marines must move through a city containing unknown areas (Figure 12.1). As the marines move into new areas, they learn the layout of the new area (large and clear, large and crowded with obstructions, limited sight corridors) and the actors in that area (civilians, enemy fire teams, snipers, enemy vehicles, etc.). Changes may occur in the environment, either structurally (an exit point becoming blocked, a wall being destroyed) or situationally (opposing force gaining high ground, loss of flanking support).

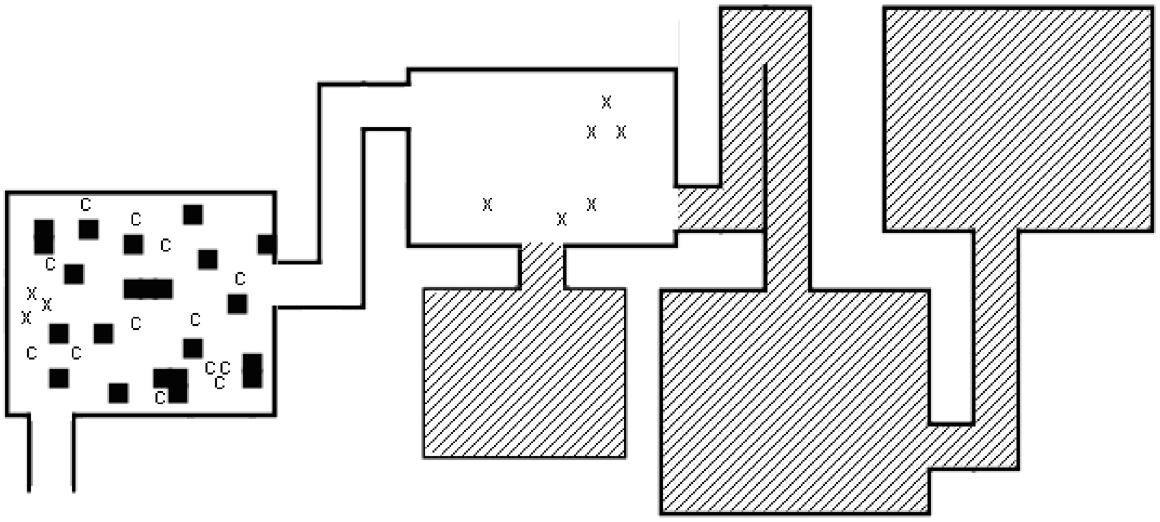


Figure 12.1. Squad of Soldiers Moves Through a City Center. From left to right: Market place filled with civilians (C), enemies (X) and crates; Open square with six enemies; Remaining areas (gray) have not been explored, contents unknown.

12.2.11 Transfer Learning

Transfer learning and generalization refer to the ability to leverage what has been learned solving one type of problem to help solve new ones. Tower defense is an excellent domain for studying this ability as solutions do not work across maps but the skills learned in building solutions do. There are many issues related to this that deserve further study.

Distance. Vector TD had range and power boosters that increased the range and power of all towers in range by 25%, with effects from multiple towers accumulating. How would the strategies and representations learned in GopherTD transfer to this situation? What if the range or power level of a tower was changed? What if we added a new type of tower that damaged all creeps in range, not just one? What if, when a creep was destroyed, it spawned two smaller, weaker creeps that also had to be destroyed? What if the map had no walls and the subject could put the towers anywhere, thus creating their own mazes?

There are many types of tower defense games, which themselves are components of real time strategy games which are approximations of real world military situations. It has been shown that humans are poor at far transfer (transferring skills to domains that appear dissimilar by virtue of having few surface features in common), but it is not clear how well they do in near transfer and how the degree of distance affects or is correlated to transfer or transfer performance.

Integration. Most studies of transfer learning assume that skills either are not transferred or, if they are transferred, can solve a problem. This is often not the case in the real world. Just as problem solving in tower defense was shown to use compositional reasoning, building solutions from sets of strategies, problem solving in other domains likely uses sets of strategies, some of which can be transferred from other domains and some of which cannot. Consider the simple case of a tower defense game that has the same types of towers as GopherTD plus one new type of tower. Many maps cannot be solved without using a strategy that leverages the capabilities of the new tower. Strategies that were valid in GopherTD are helpful but insufficient. Presumably, one would combine the relevant strategies from GopherTD with new strategies that are learned in the new domain rather than develop a new set of strategies or rely exclusively on GopherTD strategy sets. The issue of how known strategies are transferred to a new domain and integrated with a different set of strategies deserves further study.

Negative Transfer. How does transfer learning help solve novel problems? From our study it appeared that it helped by pruning the hypothesis space. For example, subjects that had played other tower defense games had learned that slowing towers were important to winning. When they were presented functionally similar towers in GopherTD, they immediately narrowed their focus to strategies involving slowing towers.

If transfer learning works by focusing the problem solver's attention to a certain area, it has the potential to harm performance by focusing the problem solver's attention to the wrong area. Early in our study we encountered a person who performed no better than chance despite having significant domain experience. We learned that in the tower defense game they normally played, slowing towers were weak and inferior to alternate types of towers. They learned that they should put all their resources into offensive towers and avoid slowing towers. They transferred this strategy to GopherTD, where it is very difficult to do well without slowing towers. They invested considerable effort into making their power-focused strategies work without ever considering a strategy that employed slowing towers.

Transfer learning does not always help. Negative transfer can actually harm performance. Further study is needed to determine how negative transfer occurs, under what conditions it occurs and measures for determining when transfer does and does not make sense.

12.2.12 Strategy Inference

Our work in tower defense originally focused on strategy inference. Given a set of solutions and the assumption that the agent that created them used a consistent set of strategies, could we determine the set of strategies the problem solver used to solve a given problem?

Our initial question was whether such a capability was possible. To answer this, we gave a domain expert a set of solutions created by one person and asked them to generate the solution they believed that person would generate on a new problem. The results were surprisingly good.

The primary process he followed was to look at each tower and ask himself why the problem solver had placed the tower there. What were they attempting to do? He attempted to recreate the problem solver's goal and intention.

It is common for multiple strategies to recommend the same position. For any given map, several strategies might seem equally plausible. The set of maps was needed to determine which of these strategies held across all maps in the solution set.

For decisions he could not explain in terms of goal, he looked to see if their decisions were consistent in some way. For example, a tower might always be placed in the center of the map or by the entrance.

In order for the first process (determine intention) to work, the inference maker must understand the task goals and the effects of placement decisions. In order for the second process (narrow down candidate strategies) to work, the inference maker must know which strategies exist. In order for the third (look for placement regularities) to work, the inference maker must know the types of placement anchors and, preferably, placement targets and relationships. All of these are within the realm of possibility and suggest that the inference task, given sufficient domain knowledge, is tractable, but further work is needed to prove this.

The value to be gained from inferring an agent's strategies includes opponent modeling, simulation, user adaptation and data analysis. The following sections give more detailed uses for this ability.

12.2.13 User-Centered Procedural Content Generation and Adaptation

There are many domains in which someone must create content for an end user. This is normally done by humans but a human-centered generation approach has three significant disadvantages. First, content generation is expensive. Second, the content must be general to address all users rather than any one specific user. Third, the content is fixed at creation and cannot change over time. Computer systems can solve all of these problems but are not commonly used because the quality of their output is poor. For many domains, understanding how spatial properties relate to affordances and affordances relate to the success of a strategy can help design and adapt content. We believe that the techniques discussed in this research can help create computer systems capable of generating and adapting content to a specific user while ensuring an acceptable level of quality. We will discuss video games, simulations and educational systems here but these ideas apply to a variety of domains.

Many procedural content generation systems (Yannakakis and Togelius 2011) are designed to auto-generate maps and levels (maps plus other content such as non-player characters and quests). Some of these systems are able to generate content tailored to a user's abilities and interests. If the system understood the types of strategies a user employed, the system could create content designed to make those strategies more useful, thus enhancing player satisfaction. Alternately, the system could create content designed to make those strategies less helpful, pushing the user to explore new approaches. In this work we mentioned that some levels have affordances so obvious that they cause subjects to learn new strategies. A procedural content generation system could be used to determine which strategies a user is employing and then teach them strategies they haven't yet learned or help them polish their use of the strategies that they already know.

Automated AI "directors" (Snowdon and Oikonomou 2011) or "virtual dungeon masters" customize a game, including difficulty level, at run time to make the game more enjoyable. Making something easier or harder is more difficult than one might think, with many games in the past having failed quite spectacularly, either because they didn't understand the impact a change had or because the change they made created opponents that were behaviorally implausible. Creating an appropriate difficulty level is often a combination of gut feel, user testing and, post-release, customer complaints, all of which come at a cost. If one understood which types of maps or situations a user was good and bad at due to an understanding of the strategies they used and the relationship between strategies and maps, the AI director could select maps that were easy or difficult for the specific player at that point in their skill development.

Designing maps that are enjoyable is another common and difficult task. The relationship between difficulty and enjoyment is unclear. In previous, unpublished work we showed that the difficulty of an opponent AI in a strategy game was positively correlated with realism but was uncorrelated with enjoyment. We did not measure enjoyment in the experiments presented here but this did not prevent subjects from making sure we knew which maps they did not enjoy. Maps where the subject's strategies were ineffective and the affordance they needed to exploit was not easy to spot frustrated and upset subjects, who believed the maps to be poorly designed and impossible to solve. Subjects who knew the appropriate strategies, but recognized that others might not do so, felt clever.

12.2.14 Learning by Demonstration

One topic of particular interest to the author is scalable authoring systems. In games, it is common for an AI programmer to develop three agents, one for the easy, normal and hard difficulty levels. There are several problems with this.

First, it is not only difficult to determine how to make something more or less difficult, it is contextually dependent. An opponent that is hard for one person to defeat might be easy for another.

Second, common methods of difficulty adjustment frequently lead to behaviorally implausible behavior. As an extreme example, the game Sharky's Pool Hall determined difficulty by modifying the probability of the opponent making a shot. An easy opponent might only make 30% of shots while a hard one might make 95%. However, lacking an ability to judge the difficulty of a shot, an easy opponent might make an extremely difficult shot that should be beyond their capabilities while a difficult player might miss an easy shot that a good player would never miss.

Third, the results are uninteresting. Opponents lack character. This is a fundamental difference between playing an AI opponent and playing online against human opponents.

One application of this work that we believe would be interesting is using strategy inference as an authoring method. Rather than an AI programmer manually creating three identical opponents that differ only in their explicitly assigned difficulty level, a group of game designers could play the game and a data mining process could extract each person's strategy set. These strategy sets would represent a data source for a simulation agent that could behave like any person whose strategies had been captured. Rather than shipping a game (simulation, training aid, etc.) with three difficulty levels, they could ship a game containing a dozen virtual opponents. The difficulty of each opponent would depend on the strategies used by the player, meaning opponent Alice would be difficult for some players and easy for others, leaving the player to determine for themselves which agents are most challenging and/or enjoyable to play against.

12.2.15 Simulation Validation

Strategy inference, user adaptation and learning by demonstration all involve simulating an existing agent rather than attempting to build an optimal agent. In our own work on simulation we were able to create agents that we believe properly simulated the people being modeled but we realized that we had no objective way to measure the quality of the simulation. Simulation was difficult for three reasons.

Comparison. Given two solutions, how does one compute the degree of similarity? In this domain, one possibility is to measure the sum of Manhattan differences – if the tower on one solution is at position (10,10) and the corresponding tower on the other solution is at (12,12), there is a four tile difference. While quantifiable, these summed differences didn't match the opinion of a domain expert we asked to judge the similarity of pairs of solutions.

One reason why is that consistency of direction and therefore intra-solution gap is ignored. Imagine the situation where two towers are next to each other on the base map. On a second map, those two towers each shifted right one tile. On a third, one tower moved right one tile and the other moved left one tile. The difference for both maps is two tiles but the second is qualitatively similar to the base map while the third is not.

Another reason the sum of differences approach doesn't work is that two positions can be the same in justification by significantly different in position. For example, imagine that the map is a 10x10 square and the strategy is to place a tower on a corner. A tower is placed on the southwest corner of the base map. The tower is placed on the northeast corner on the second map and two tiles to the east of the southwest corner on the third map. The differences are 20 and 2 respectively but the second map is qualitatively the same as the base map while the third is not.

Inconsistency. During testing some subjects attempted to re-use solutions they had created in training. We observed that subjects could normally qualitatively recreate their solutions but not quantitatively. They remembered the affordances present and the strategies they used but not the precise location of the tower. In a different set of experiments, subjects attempted to perfectly recreate a solution they had just tested and were frequently unable to do so. Subjects were often consistent in their strategy use but not in precise tower placement.

Drift. We noticed that subjects often created solutions that varied slightly from what their self-described strategy use predicted. In interviews subjects often attributed this to curiosity; they knew the set of strategies they wanted to use but would occasionally put a tower somewhere else just to see what happened. If a change worked well and they had a theoretical explanation for what they did (since the precise movement would have no analogue on a new problem), they continued with that move and varied something else.

Attempting to compare a simulation agent to a human agent becomes a difficult task as the human is a moving target. In tests we noticed that our simulation agents resembled a "cleaner" version of the subjects being modeled. Deliberately varying our actions - and then perceptually noticing the new opportunities or problems that such variation has generated – is a pervasive form of affordance-driven learning that humans undertake in an ongoing cycle of perception and action, concreteness and abstraction.

References

Aho, Alfred V, John E Hopcroft, and Jeffrey D Ullman. 1974. *The Design and Analysis of Algorithms*. Addison-Wesley Reading.

Allen, James F. 1983. "Maintaining Knowledge about Temporal Intervals." *Communications of the ACM* 26 (11): 832–43.

Amthauer, R. 1973. *Intelligenz-Struktur-Test 70*. Gottingen, Germany: Hogrefe.

Anderson, John R. 2007. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Anderson, John R., Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. 2004. "An Integrated Theory of the Mind." *Psychological Review* 111: 1036–60.

Arentze, Theo A., Benedict G. C. Dellaert, and Harry J. P. Timmermans. 2008. "Modeling and Measuring Individuals' Mental Representations of Complex Spatio-Temporal Decision Problems." *Environment and Behavior* 40: 843–69.

Astur, Robert S, Maria L Ortiz, and Robert J Sutherland. 1998. "A Characterization of Performance by Men and Women in a Virtual Morris Water Task:: A Large and Reliable Sex Difference." *Behavioural Brain Research* 93 (1): 185–90.

Baenninger, Maryann, and Nora Newcombe. 1989. "The Role of Experience in Spatial Test Performance: A Meta-Analysis." *Sex Roles* 20 (5-6): 327–44.

Ball, Karlene K, Bettina L Beard, Daniel L Roenker, Richard L Miller, and David S Griggs. 1988. "Age and Visual Search: Expanding the Useful Field of View." *JOSA A* 5 (12): 2210–19.

Barsalou, Lawrence W. 2008. "Grounded Cognition." *Annu. Rev. Psychol.* 59: 617–45.

Beek, Peter van, and Dennis W. Manchak. 1996. "The Design and Experimental Analysis of Algorithms for Temporal Reasoning." *CoRR* cs.AI/9601101. http://arxiv.org/abs/cs.AI/9601101.

Behr, Merlyn J, Ipke Wachsmuth, Thomas R Post, and Richard Lesh. 1984. "Order and Equivalence of Rational Numbers: A Clinical Teaching Experiment." *Journal for Research in Mathematics Education*, 323–41.

Bennett, Brandon. 1994. "Spatial Reasoning with Propositional Logics." *KR* 94: 51–62.

———. 1998. "Determining Consistency of Topological Relations." *Constraints* 3 (2-3): 213–25.

Berenbaum, Sheri A. 1999. "Effects of Early Androgens on Sex-Typed Activities and Interests in Adolescents with Congenital Adrenal Hyperplasia." *Hormones and Behavior* 35 (1): 102–10.

Berenbaum, Sheri A, and Melissa Hines. 1992. "Early Androgens Are Related to Childhood Sex-Typed Toy Preferences." *Psychological Science* 3 (3): 203–6.

Bertel, Sven, and Alex Kirlik. 2010. "Fast and Frugal Heuristics." *Wiley Encyclopedia of Operations Research and Management Science*.

Bodirsky, Manuel, and Stefan Wölfl. 2011. "RCC8 Is Polynomial on Networks of Bounded Treewidth." In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence-Volume Volume Two*, 756–61.

Boroditsky, L. 2000. "Metaphoric Structuring: Understanding Time through Spatial Metaphors." *Cognition* 75 (1): 1–28. doi:10.1016/S0010-0277(99)00073-6. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2A4D381CBF66F4FA7441A9C5DDB80AA0?doi=10.1.1.11.5402&rep=rep1&type=pdf.

Bredeweg, Bert, and Peter Struss. 2004. “Current Topics in Qualitative Reasoning.” *AI Mag.* 24 (4): 13–16. http://dl.acm.org/citation.cfm?id=973340.973343.

Brehmer, Berndt. 1992. “Dynamic Decision Making: Human Control of Complex Systems.” *Acta Psychologica* 81 (3): 211–41.

Buro, Michael. 2003. “Real-Time Strategy Games: A New AI Research Challenge.” In *Int’l Joint Conf. on Artifical Intelligence*, 1534–35.

Cabalar, Pedro, and Paulo Santos. 2006. “Strings and Holes: An Exercise on Spatial Reasoning.” In *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, 419–29. Springer.

Cabalar, Pedro, and Paulo E. Santos. 2011. “Formalising the Fisherman’s Folly Puzzle.” *Artif. Intell.* 175 (1): 346–77. doi:10.1016/j.artint.2010.04.004. http://www.dc.fi.udc.es/ cabalar/fishermansAIJ.pdf.

Canas, Jose, Jose Quesada, Adoración Antolí, and Inmaculada Fajardo. 2003. “Cognitive Flexibility and Adaptability to Environmental Changes in Dynamic Complex Problem-Solving Tasks.” *Ergonomics* 46 (5): 482–501.

Casasanto, Daniel, and Lera Boroditsky. 2008. “Time in the Mind: Using Space to Think about Time.” *Cognition* 106: 579–93.

Chai, Xiaoqian J, and Lucia F Jacobs. 2009. “Sex Differences in Directional Cue Use in a Virtual Landscape.” *Behavioral Neuroscience* 123 (2): 276.

Cherrier, MM, S Asthana, S Plymate, L Baker, AM Matsumoto, E Peskind, MA Raskind, et al. 2001. “Testosterone Supplementation Improves Spatial and Verbal Memory in Healthy Older Men.” *Neurology* 57 (1): 80–88.

Clementini, Eliseo, Paolino Di Felice, and Daniel Hernández. 1997. “Qualitative Representation of Positional Information.” *Artificial Intelligence* 95 (2): 317–56.

Cohn, Anthony G, and Jochen Renz. 2008. “Qualitative Spatial Representation and Reasoning.” *Foundations of Artificial Intelligence* 3: 551–96.

Cohn, Anthony G., and Shyamanta M. Hazarika. 2001. “Qualitative Spatial Representation and Reasoning: An Overview.” *Fundamenta Informaticae* 46 (1): 1–29.

Cohn, Anthony G., David A. Randell, and Zhan Cui. 1994. "Taxonomies of Logically Defined Qualitative Spatial Relations." In *IN N. GUARINO AND R. POLI (EDS), FORMAL ONTOLOGY IN CONCEPTUAL ANALYSIS AND KNOWLEDGE REPRESENTATION*, 831–46. Kluwer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.1360&rep=rep1&type=pdf.

Collins, David W, and Doreen Kimura. 1997. "A Large Sex Difference on a Two-Dimensional Mental Rotation Task." *Behavioral Neuroscience* 111 (4): 845.

Collins, Gregg. 1987. "Plan Creation: Using Strategies as Blueprints". Yale.

Colom, Roberto, M^{a} José Contreras, Juan Botella, and José Santacreu. 2002. "Vehicles of Spatial Ability." *Personality and Individual Differences* 32 (5): 903–12.

Contreras, María José, Roberto Colom, Pei C Shih, María Jesús Álava, and José Santacreu. 2001. "Dynamic Spatial Performance: Sex and Educational Differences." *Personality and Individual Differences* 30 (1): 117–26.

Costantini, Marcello, Ettore Ambrosini, Claudia Scorolli, and Anna M Borghi. 2011. "When Objects Are close to Me: Affordances in the Peripersonal Space." *Psychonomic Bulletin & Review* 18 (2): 302–8.

Dean, Thomas L, and Drew V McDermott. 1987. "Temporal Data Base Management." *Artificial Intelligence* 32 (1): 1–55.

Dechter, Rina, Itay Meiri, and Judea Pearl. 1991. "Temporal Constraint Networks." *Artificial Intelligence* 49 (1): 61–95.

Detterman, D.K., and R.J. Sternberg. 1993. *Transfer on Trial: Intelligence, Cognition, and Instruction*. Ablex Pub. Corp. http://books.google.com/books?id=AYYOAQAAMAAJ.

Dimov, Georgi, and Dimiter Vakarelov. 2006. "Contact Algebras and Region-Based Theory of Space: A Proximity Approach-I." *Fundamenta Informaticae* 74 (2): 209–49.

Duan, Ling-Yu, Jie Chen, Rongrong Ji, Tiejun Huang, and Wen Gao. 2013. “Learning Compact Visual Descriptors for Low Bit Rate Mobile Landmark Search.” *AI Magazine* 34 (2): 67.

Duncan, John. 1984. “Selective Attention and the Organization of Visual Information.” *Journal of Experimental Psychology: General* 113 (4): 501.

Duncker, Karl, and Lynne S Lees. 1945. “On Problem-Solving.” *Psychological Monographs* 58 (5): i.

Dylla, Frank. 2008. *An Agent Control Perspective on Qualitative Spatial Reasoning: Towards More Intuitive Spatial Agent Development*. Dissertationen Zur Künstlichen Intelligenz. IOS Press. http://books.google.com/books?id=sn50zfXY3oUC.

Dylla, Frank, and Reinhard Moratz. 2005. “Exploiting Qualitative Spatial Neighborhoods in the Situation Calculus.” In *Spatial Cognition IV. Reasoning, Action, Interaction*, 304–22. Springer.

Edwards, Ward. 1962. “Dynamic Decision Theory and Probabilistic Information Processings.” *Human Factors: The Journal of the Human Factors and Ergonomics Society* 4 (2): 59–74.

Egenhofer, Max J. 1994. “Deriving the Composition of Binary Topological Relations.” *Journal of Visual Languages & Computing* 5 (2): 133–49.

Egenhofer, Max J, and Robert D Franzosa. 1991. “Point-Set Topological Spatial Relations.” *International Journal of Geographical Information System* 5 (2): 161–74.

Ekstrom, Ruth B, John W French, Harry Horace Harman, and David Dermen. 1976. *Manual for Kit of Factor-Referenced Cognitive Tests*. Princeton, NJ: Educational testing service.

Evans, Thomas G. 1964. “A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions”. DTIC Document.

———. 1968. “A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions.” In *Semantic Information Processing*, edited by Marvin Minsky, 271–35. MIT Press.

Feng, Jing, Ian Spence, and Jay Pratt. 2007. “Playing an Action Video Game Reduces Gender Differences in Spatial Cognition.” *Psychological Science* 18 (10): 850–55.

Forbus, Kenneth D, James V Mahoney, and Kevin Dill. 2002. “How Qualitative Spatial Reasoning Can Improve Strategy Game AIs.” *Intelligent Systems, IEEE* 17 (4): 25–30.

Forbus, Kenneth D, Paul Nielsen, and Boi Faltings. 1991. “Qualitative Spatial Reasoning: The CLOCK Project.” *Artificial Intelligence* 51 (1): 417–71.

Frank, Andrew U. 1991. “Qualitative Spatial Reasoning with Cardinal Directions.” In *7. Österreichische Artificial-Intelligence-Tagung/Seventh Austrian Conference on Artificial Intelligence*, 157–67.

Freksa, Christian. 1992a. “Using Orientation Information for Qualitative Spatial Reasoning.” In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space: International Conference GIS–From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, Pisa, Italy, September 21-23, 1992: Proceedings*, edited by Andrew U Frank, Irene Campari, and Ubaldo Formentini. Pisa, Italy: Springer-Verlag.

———. 1992b. “Temporal Reasoning Based on Semi-Intervals.” *Artificial Intelligence* 54 (1–2): 199 – 227. doi:http://dx.doi.org/10.1016/0004-3702(92)90090-K. http://www.sciencedirect.com/science/article/pii/000437029290090K.

French, John W, Ruth B Ekstrom, and Leighton A Price. 1963. “Manual for Kit of Reference Tests for Cognitive Factors (revised 1963)”. DTIC Document.

Frensch, Peter A, and Joachim Funke. 1995. *Complex Problem Solving: The European Perspective*. Psychology Press.

Frommberger, Lutz. 2008. “Learning to Behave in Space: A Qualitative Spatial Representation for Robot Navigation with Reinforcement Learning.”

International Journal on Artificial Intelligence Tools 17 (03): 465–82. http://www.aussagekraft.de/files/Frommberger-IJAIT08.pdf.

Fuhrman, Orly, and Lera Boroditsky. 2010. "Cross-Cultural Differences in Mental Representations of Time: Evidence From an Implicit Nonlinguistic Task." *Cognitive Science* 34 (8): 1430–51.

Fuhrman, Orly, Kelly McCormick, Eva Chen, Heidi Jiang, Dingfang Shu, Shuaimei Mao, and Lera Boroditsky. 2011. "How Linguistic and Cultural Forces Shape Conceptions of Time: English and Mandarin Time in 3D." *Cognitive Science* 35: 1305–28.

Galea, Liisa AM, and Doreen Kimura. 1993. "Sex Differences in Route-Learning." *Personality and Individual Differences* 14 (1): 53–65.

Gaschler, Robert, and Peter A Frensch. 2007. "Is Information Reduction an Item-Specific or an Item-General Process?" *International Journal of Psychology* 42 (4): 218–28.

Gaulin, Steven JC, and Randall W FitzGerald. 1986. "Sex Differences in Spatial Ability: An Evolutionary Hypothesis and Test." *American Naturalist*, 74–88.

Gerevini, Alfonso, and Bernhard Nebel. 2002. "Qualitative Spatio-Temporal Reasoning with RCC-8 and Allen's Interval Calculus: Computational Complexity." In *ECAI*, 2:312–16.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. Boston, Mass.: Houghton Mifflin.

Gigerenzer, Gerd. 1991. "How to Make Cognitive Illusions Disappear: Beyond 'Heuristics and Biases.'" *European Review of Social Psychology* 2 (1): 83–115. doi:10.1080/14792779143000033. http://www.tandfonline.com/doi/abs/10.1080/14792779143000033.

———. 1996. "On Narrow Norms and Vague Heuristics: A Reply to Kahneman and Tversky." *Psychological Review* 103 (3): 592–96.

———. 2004. "Fast and Frugal Heuristics: The Tools of Bounded Rationality." *Blackwell Handbook of Judgment and Decision Making*, 62–88.

Gigerenzer, Gerd, Wolfgang Gaissmaier, Elke Kurz-Milcke, Lisa M. Schwartz, and Steven Woloshin. 2008. “Helping Doctors and Patients Make Sense of Health Statistics.” *PSYCHOLOGICAL SCIENCE IN THE PUBLIC INTEREST* 8 (2): 53–96. http://www.psychologicalscience.org/journals/pspi/pspi_8_2_article.pdf.

Gigerenzer, Gerd, and Daniel G Goldstein. 1996. “Reasoning the Fast and Frugal Way: Models of Bounded Rationality.” *Psychological Review* 103 (4): 650.

Gigerenzer, Gerd, and Peter M Todd. 1999a. *Simple Heuristics That Make Us Smart*. Oxford University Press New York.

———. 1999b. “Fast and Frugal Heuristics: The Adaptive Toolbox.”

Goldstone, R., and L. W. Barsalou. 1998. “Reuniting Perception and Conception.” *Cognition* 65: 231–62.

Gonzalez, C., P. Vanyukov, and M. K. Martin. 2005. “The Use of Microworlds to Study Dynamic Decision Making.” *Computers in Human Behavior* 21: 273–86.

Goode, Natassia, and Jens F Beckmann. 2010. “You Need to Know: There Is a Causal Relationship between Structural Knowledge and Control Performance in Complex Problem Solving Tasks.” *Intelligence* 38 (3): 345–52.

Gordon, Andrew S. 2001. “Playing Chess with Machiavelli: Improving Interactive Entertainment with Explicit Strategies.” In *The 2001 Spring Symposium on Artificial Intelligence and Interactive Entertainment*, 26–28.

Gordon, Andrew S. 2004a. “The Representation of Planning Strategies.” *Artificial Intelligence* 153: 287–305.

———. 2004b. *Strategy Representation: An Analysis of Planning Knowledge*. Lawrence Erlbaum Associates.

Gorniak, Peter, and Deb Roy. 2007. “Situated Language Understanding as Filtering Perceived Affordances.” *Cognitive Science* 31 (2): 197–231.

Gottfried, Björn. 2004. “Reasoning about Intervals in Two Dimensions.” In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 6:5324–32.

Gouchie, Catherine, and Doreen Kimura. 1991. “The Relationship between Testosterone Levels and Cognitive Ability Patterns.” *Psychoneuroendocrinology* 16 (4): 323–34.

Grigni, Michelangelo, Dimitris Papadias, and Christos Papadimitriou. 1995. “Topological Inference.” In *IJCAI (1)*, 901–7.

Grön, Georg, Arthur P Wunderlich, Manfred Spitzer, Reinhard Tomczak, and Matthias W Riepe. 2000. “Brain Activation during Human Navigation: Gender-Different Neural Networks as Substrate of Performance.” *Nature Neuroscience* 3 (4): 404–8.

Guidi, John N, and Thomas H Roderick. 1993. “Inference of Order in Genetic Systems.” In *ISMB*, 163–71.

Guilford, JP. 1956. “The Guilford-Zimmerman Aptitude Survey.” *The Personnel and Guidance Journal* 35 (4): 219–23.

Guilford, JP, and Wayne S Zimmerman. 1948. “The Guilford-Zimmerman Aptitude Survey.” *Journal of Applied Psychology* 32 (1): 24.

Hammond, Tracy. 2007. “LADDER: A Perceptually-Based Language to Simplify Sketch Recognition User Interfaces Development”. MIT.

Hammond, Tracy, and Randall Davis. 2006. “LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition.” In *ACM SIGGRAPH 2006 Courses*, 27.

Harris, Lauren Julius. 1978. *Sex Differences in Spatial Ability: Possible Environmental, Genetic, and Neurological Factors*. Asymmetrical Junction of the brain. Cambridge: Cambridge University Press.

Haskell, Robert E. 2000. *Transfer of Learning: Cognition and Instruction*. Academic Press.

Hausmann, Markus, Ditte Slabbekoorn, Stephanie HM Van Goozen, Peggy T Cohen-Kettenis, and Onur Güntürkün. 2000. “Sex Hormones Affect Spatial Abilities during the Menstrual Cycle.” *Behavioral Neuroscience* 114 (6): 1245.

Hernández, Daniel, Eliseo Clementini, and Paolino Felice. 1995. "Qualitative Distances." In *Spatial Information Theory A Theoretical Basis for GIS*, edited by AndrewU. Frank and Werner Kuhn, 988:45–57. Lecture Notes in Computer Science. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-60392-1_4.

Holding, Carol S, and Dennis H Holding. 1989. "Acquisition of Route Network Knowledge by Males and Females." *The Journal of General Psychology* 116 (1): 29–41.

Houghton, George, and Steven P Tipper. 1994. "A Model of Inhibitory Mechanisms in Selective Attention."

Hué, Julien, Mariette Sérayet, Pierre Drap, Odile Papini, and Eric Würbel. 2011. "Underwater Archaeological 3D Surveys Validation within the Removed Sets Framework." In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 663–74. Springer.

Isli, Amar. 2004. "Combining Cardinal Direction Relations and Other Orientation Relations in QSR." In *Proc. of 8th International Symposium on Artificial Intelligence and Mathematics (AI&M'04).*

John Raven, John Carlyle Raven, and John Court. 2004. *Manual for Raven's Progressive Matrices and Vocabulary Scales*. San Antonio, TX: Harcourt Assessment.

Johnson, Wendy, and Thomas J Bouchard Jr. 2007. "Sex Differences in Mental Abilities:< I> G</i> Masks the Dimensions on Which They Lie." *Intelligence* 35 (1): 23–39.

Jonsson, Peter, and Thomas Drakengren. 1997. "A Complete Classification of Tractability in RCC-5." *arXiv Preprint cs/9706102*.

Kahneman, Daniel, and Amos Tversky. 1996. "On the Reality of Cognitive Illusions." *Psychological Review* 103 (3): 582–91.

Kaplunova, Alissa, Volker Haarslev, and Ralf Möller. 2002. "Adding Ternary Complex Roles to ALCRP (D)." In *Proceedings of the International Workshop on Description Logics (DL-2002), Toulouse, France*, 45–52.

Kautz, Henry A, and Peter B Ladkin. 1991. "Integrating Metric and Qualitative Temporal Reasoning." In *AAAI*, 91:241–46.

Kimura, Doreen. 1992. "Sex Differences in the Brain." *Scientific American* 267 (3): 118–25.

Kimura, Doreen, and Elizabeth Hampson. 1994. "Cognitive Pattern in Men and Women Is Influenced by Fluctuations in Sex Hormones." *Current Directions in Psychological Science* 3 (2): 57–61.

Klein, Gary A. 1998. *Sources of Power: How People Make Decisions*. MIT press.

Kogan, Nathan, and Karen Dorros. 1978. "Sex Differences in Risk Taking and Its Attribution." *Sex Roles* 4 (5): 755–65.

Kozhevnikov, Maria, and Mary Hegarty. 2001. "A Dissociation between Object Manipulation Spatial Ability and Spatial Orientation Ability." *Memory & Cognition* 29 (5): 745–56.

Kreutzmann, Arne, and Diedrich Wolter. 2011. "Physical Puzzles—Challenging Spatio-Temporal Configuration Problems." In *IJCAI-2011 Workshop Benchmarks and Applications of Spatial Reasoning*, 21.

Ladkin, Peter B, and Roger D Maddux. 1987. *The Algebra of Convex Time Intervals*. Citeseer.

Laird, John E, Allen Newell, and Paul S Rosenbloom. 1987. "Soar: An Architecture for General Intelligence." *Artificial Intelligence* 33 (1): 1–64.

Lathrop, Scott D, and John E Laird. 2007. "Towards Incorporating Visual Imagery into a Cognitive Architecture." In *Proceedings of the Eighth International Conference on Cognitive Modeling*.

———. 2009. "Extending Cognitive Architectures with Mental Imagery." In *Proceedings of the Second Conference on Artificial General Intelligence*.

Lave, J. 1988. *Cognition in Practice: Mind, Mathematics and Culture in Everyday Life*. Cambridge University Press. http://books.google.com/books?id=n6eiH3iPVKYC.

Lawton, Carol A. 1994. "Gender Differences in Way-Finding Strategies: Relationship to Spatial Ability and Spatial Anxiety." *Sex Roles* 30 (11-12): 765–79.

Lebiere, Christian, Peter Pirolli, Robert Thomson, Jaehyon Paik, Matthew Rutledge-Taylor, James Staszewski, and John R Anderson. 2013. “A Functional Model of Sensemaking in a Neurocognitive Architecture.” *Computational Intelligence and Neuroscience* 2013.

Lee, Jae Hee, and Diedrich Wolter. 2011. “A New Perspective on Reasoning with Qualitative Spatial Knowledge.” In *IJCAI-2011 Workshop Benchmarks and Applications of Spatial Reasoning*, 3.

Levy, Lauren J, Robert S Astur, and Karyn M Frick. 2005. “Men and Women Differ in Object Memory but Not Performance of a Virtual Radial Maze.” *Behavioral Neuroscience* 119 (4): 853.

Lewis, Clayton H. 1982. “Using the Thinking Aloud Method In Cognitive Interface Design”. IBM RC-9265. Yorktown Heights, NY: IBM.

Li, Jason Jingshi, Jinbo Huang, and Jochen Renz. 2009. “A Divide-and-Conquer Approach for Solving Interval Algebra Networks.” In *IJCAI*, 2009:572–77.

Li, Jason Jingshi, Jochen Renz, and Li Sanjiang. 2008. “Combining Binary Constraint Networks in Qualitative Reasoning.”

Li, Sanjiang, and Mingsheng Ying. 2003a. “Extensionality of the RCC8 Composition Table.” *Fundamenta Informaticae* 55 (3): 363–85.

———. 2003b. “Region Connection Calculus: Its Models and Composition Table.” *Artificial Intelligence* 145 (1): 121–46.

Linn, Marcia C, and Anne C Petersen. 1985. “Emergence and Characterization of Sex Differences in Spatial Ability: A Meta-Analysis.” *Child Development*, 1479–98.

Lipshitz, Raanan, Gary Klein, Judith Orasanu, and Eduardo Salas. 2001. “Taking Stock of Naturalistic Decision Making.” *Journal of Behavioral Decision Making* 14 (5): 331–52.

Lovett, Andrew, Kenneth Forbus, and Jeffrey Usher. 2007. “Analogy with Qualitative Spatial Representations Can Simulate Solving Raven’s Progressive Matrices.” In *Proceedings of the 29th Annual Conference of the Cognitive Society*.

Lovett, Andrew, Forbus, Kenneth, and Usher, Jeffrey. 2010. “A Structure-Mapping Model of Raven’s Progressive Matrices.” In *Proceedings of Cognitive Science 2010.*

Lücke, Dominik, and Till Mossakowski. 2010. “A Much Better Polynomial Time Approximation of Consistency in the LR Calculus.” In *STAIRS*, 175–85.

Mackworth, Alan K. 1977. “Consistency in Networks of Relations.” *Artificial Intelligence* 8 (1): 99–118.

Mandler, Jean M. 2012. “On the Spatial Foundations of the Conceptual System and Its Enrichment.” *Cognitive Science* 36 (3): 421–51.

Mandler, Jean Matter. 2004. *The Foundations of Mind: Origins of Conceptual Thought.* Oxford University Press New York:

Marr, M Jackson. 2003. “The Stitching and the Unstitching: What Can Behavior Analysis Have to Say about Creativity?” *The Behavior Analyst* 26 (1): 15.

Marulis, LM, LL Lui, CM Warren, DH Uttal, and NS Newcombe. 2007. “Effects of Training or Experience on Spatial Cognition in Children and Adults: A Meta-Analysis.” In *Annual Meeting of the American Educational Research Association, Chicago, IL.*

Masters, Mary Soares, and Barbara Sanders. 1993. “Is the Gender Difference in Mental Rotation Disappearing?” *Behavior Genetics* 23 (4): 337–41.

McCarthy, John. 1990. “Chess as the Drosophila of AI.” In *Computers, Chess, and Cognition*, 227–37. Springer.

———. 1998. “Partial Formalizations and the Lemmings Game”. Technical report, Stanford University, Formal Reasoning Group.

McKeever, Walter F, Deborah A Rich, Richard A Deyo, and Robert L Conner. 1987. “Androgens and Spatial Ability: Failure to Find a Relationship between Testosterone and Ability Measures.” *Bulletin of the Psychonomic Society.*

Meathrel, Richard, and Antony Galton. 2001. “A Hierarchy of Boundary-Based Shape Descriptors.” In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 17:1359–64.

Miles, Lynden K., Ewa Betka, Louise F. Pendry, and C. Neil Macrae. 2010. "Mapping Temporal Constructs: Actions Reveal That Time Is a Place." *Quarterly Journal of Experimental Psychology* 63 (11): 2113–19. http://www.biomedsearch.com/nih/Mapping-temporal-constructs-Actions-reveal/20981632.html.

Miller, Leon K, and Viana Santoni. 1986. "Sex Differences in Spatial Abilities: Strategic and Experiential Correlates." *Acta Psychologica* 62 (3): 225–35.

Moffat, Scott D, Elizabeth Hampson, and Maria Hatzipantelis. 1998. "Navigation in a 'virtual' Maze: Sex Differences and Correlation with Psychometric Measures of Spatial Ability in Humans." *Evolution and Human Behavior* 19 (2): 73–87.

Montanari, Ugo. 1974. "Networks of Constraints: Fundamental Properties and Applications to Picture Processing." *Information Sciences* 7: 95–132.

Moratz, Reinhard. 2006. "Representing Relative Direction as a Binary Relation of Oriented Points." In *ECAI*, 6:407–11.

Moratz, Reinhard, Frank Dylla, and Lutz Frommberger. 2005. "A Relative Orientation Algebra with Adjustable Granularity." In *Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments (IJCAI 05)*, 61–70.

Moratz, Reinhard, Jochen Renz, and Diedrich Wolter. 2000. "Qualitative Spatial Reasoning about Line Segments." In *ECAI*, 234–38.

Morris, Richard GM. 1981. "Spatial Localization Does Not Require the Presence of Local Cues." *Learning and Motivation* 12 (2): 239–60.

Mossakowski, Till, Christian Maeder, and Klaus Lüttich. 2007. "The Heterogeneous Tool Set, HETS." In *Tools and Algorithms for the Construction and Analysis of Systems*, 519–22. Springer.

Norman, Donald A. 2002. *The Design of Everyday Things*. Basic books.

Novick, Laura R. 1990. "Representational Transfer in Problem Solving." *Psychological Science* 1 (2): 128–32.

Oltman, Philip K, Evelyn Raskin, Herman A Witkin, and Consulting Psychologists Press. 1971. *Group Embedded Figures Test*. Consulting Psychologists Press Palo Alto, CA.

Paik, Jaehyon, Peter Pirolli, Wei Dong, Christian Lebiere, and Robert Thomson. 2013. “An ACT-R Model of Sensemaking in Geospatial Intelligence Tasks.” *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, July.

Pellegrino, J. W., and E. Hunt. 1989. “Computer Controlled Assessment of Static and Dynamic Spatial Reasoning.” In *Testing: Theoretical and Applied Perspectives*, edited by E. F. Dillon and J. W. Pellegrino, 174–98. Praeger.

Peters, Michael, Bruno Laeng, Kerry Latham, Marla Jackson, Raghad Zaiyouna, and Chris Richardson. 1995. “A Redrawn Vandenberg and Kuse Mental Rotations Test-Different Versions and Factors That Affect Performance.” *Brain and Cognition* 28 (1): 39–58.

Peuquet, Donna J, and Zhan Ci-Xiang. 1987. “An Algorithm to Determine the Directional Relationship between Arbitrarily-Shaped Polygons in the Plane.” *Pattern Recognition* 20 (1): 65–74.

Piaget, Jean. 1927. *The Child’s Conception of Time*. Translated by Arnold Julius Pomerans. Routledge & K. Paul.

Piaget, Jean, David Elkind, and Anita Tenzer. 1967. *Six Psychological Studies*. Random House New York.

Picazo-Tadeo, Andres J, Ernest Reig-Martinez, and Francesc Hernandez-Sancho. 2005. “Directional Distance Functions and Environmental Regulation.” *Resource and Energy Economics* 27 (2): 131–42.

Pujari, Arun K., Vijaya G. Kumari, and Abdul Sattar. 1999. “INDu: An Interval & Duration Network.” In *Advanced Topics in Artificial Intelligence*, edited by Norman Foo, 1747:291–303. Lecture Notes in Computer Science. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-46695-9_25.

Ragni, Marco, and Sven Brüssow. 2011. “Human Spatial Relational Reasoning: Processing Demands, Representations, and Cognitive Model.” In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 828–33.

Ragni, Marco, Bolormaa Tseden, and Markus Knauff. 2007. “Cross-Cultural Similarities in Topological Reasoning.” In *Spatial Information Theory*, 32–46. Springer.

Randell, David A., Zhan Cui, and Anthony G. Cohn. 1992. “A Spatial Logic Based on Regions and Connection.” In *Proc. Int’l Conf. on Knowledge Representation and Reasoning*.

Reimer, Torsten, and Jörg Rieskamp. 2007. “Fast and Frugal Heuristics.” *Encyclopedia of Social Psychology, Thousand Oaks, CA: Sage*.

Renz, Jochen. 2002a. “A Canonical Model of the Region Connection Calculus.” *Journal of Applied Non-Classical Logics* 12 (3-4): 469–94. doi:10.3166/jancl.12.469-494. http://www.tandfonline.com/doi/abs/10.3166/jancl.12.469-494.

———. 2002b. *Qualitative Spatial Reasoning with Topological Information*. Springer-Verlag.

Renz, Jochen, and Gérard Ligozat. 2005. “Weak Composition for Qualitative Spatial and Temporal Reasoning.” In *Principles and Practice of Constraint Programming-CP 2005*, 534–48. Springer.

Renz, Jochen, and Bernhard Nebel. 2007. “Qualitative Spatial Reasoning Using Constraint Calculi.” In *Handbook of Spatial Logics*, 161–215. Springer.

Renz, Jochen, Reinhold Rauh, and Markus Knauff. 2000. *Towards Cognitive Adequacy of Topological Spatial Relations*. Springer.

Ruscio, John. 2003. “Comparing Bayes’s Theorem to Frequency-Based Approaches to Teaching Bayesian Reasoning.” *Teaching of Psychology* 30: 325—328.

Sanders, Barbara, Mary P Soares, and Jean M D’Aquila. 1982. “The Sex Difference on One Test of Spatial Visualization: A Nontrivial Difference.” *Child Development*, 1106–10.

Schlieder, Christoph. 1995. “Reasoning about Ordering.” In *Spatial Information Theory a Theoretical Basis for GIS*, 341–49. Springer.

Scivos, Alexander, and Bernhard Nebel. 2001. “Double-Crossing: Decidability and Computational Complexity of a Qualitative Calculus for Navigation.” In *Spatial Information Theory*, 431–46. Springer.

Sedlmeier, Peter, and Gerd Gigerenzer. 2001. “Teaching Bayesian Reasoning in Less than Two Hours.” *Journal of Experimental Psychology: General* 130 (3): 380.

Senge, Peter M. 1997. “The Fifth Discipline.” *Measuring Business Excellence* 1 (3): 46–51.

Sherry, David F, and Elizabeth Hampson. 1997. “Evolution and the Hormonal Control of Sexually-Dimorphic Spatial Abilities in Humans.” *Trends in Cognitive Sciences* 1 (2): 50–56.

Silverman, Irwin, and Marion Eals. 1992. “Sex Differences in Spatial Abilities: Evolutionary Theory and Data.” In *Portions of This Paper Were Presented at the Meetings of the International Society for Human Ethology in Binghamton, NY, Jun 1990, the Human Behavior and Evolution Society in Los Angeles, CA, Aug 1990, and the European Sociobiological Society in Prague, Czechoslovakia, Aug 1991.*

Silverman, Irwin, Don Kastuk, Jean Choi, and Krista Phillips. 1999. “Testosterone Levels and Spatial Ability in Men.” *Psychoneuroendocrinology* 24 (8): 813–22.

Simon, Herbert A. 1956. “Rational Choice and the Structure of the Environment.” *Psychological Review* 63 (2): 129.

Singley, M.K., and J.R. Anderson. 1989. *The Transfer of Cognitive Skill*. Cognitive Science Series, 9. Harvard University Press. http://books.google.com/books?id=8UrN-09ZXUsC.

Slabbekoorn, Ditte, Stephanie HM van Goozen, Jos Megens, Louis JG Gooren, and Peggy T Cohen-Kettenis. 1999. “Activating Effects of Cross-Sex Hormones on Cognitive Functioning: A Study of Short-Term and Long-Term Hormone Effects in Transsexuals.” *Psychoneuroendocrinology* 24 (4): 423–47.

Smith, Michael. 2006. “Running the Table: An AI for Computer Billiards.” In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL*

INTELLIGENCE, 21:994. http://webdocs.cs.ualberta.ca/ jonathan/PREVIOUS/Papers/Papers/pickpocket.pdf.

Smith, John P., Disessa, Andrea A., and Jeremy Roschelle. 1993. "Misconceptions Reconceived: A Constructivist Analysis of Knowledge in Transition." *The Journal of the Learning Sciences* 3 (2): 115–63. http://www.jstor.org/stable/1466679.

Snowdon, John, and Andreas Oikonomou. 2011. "Creating More Entertaining and Re-Playable Games by Dynamically Introducing and Manipulating, Static Gameplay Elements and Events." In *CGAMES*, 94–100.

Spence, Ian, Jingjie Jessica Yu, Jing Feng, and Jeff Marshman. 2009. "Women Match Men When Learning a Spatial Skill." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 35 (4): 1097.

Sweeney, Linda Booth, and John D Sterman. 2000. "Bathtub Dynamics: Initial Results of a Systems Thinking Inventory." *System Dynamics Review* 16 (4): 249–86.

Terlecki, Melissa S, Nora S Newcombe, and Michelle Little. 2008. "Durable and Generalized Effects of Spatial Experience on Mental Rotation: Gender Differences in Growth Patterns." *Applied Cognitive Psychology* 22 (7): 996–1013.

Thau, David, Shawn Bowers, and Bertram Ludäscher. 2008. "Merging Taxonomies under RCC-5 Algebraic Articulations." In *Proceedings of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web*, 47–54.

Thill, Serge, Daniele Caligiore, Anna M Borghi, Tom Ziemke, and Gianluca Baldassarre. 2013. "Theories and Computational Models of Affordance and Mirror Systems: An Integrative Review." *Neuroscience & Biobehavioral Reviews*.

Tomai, E, K Forbus, and J Usher. 2004. "Qualitative Spatial Reasoning for Geometric Analogies." In *Proceedings of the 18th International Qualitative Reasoning Workshop*.

Treisman, Anne, and Stephen Gormican. 1988. "Feature Analysis in Early Vision: Evidence from Search Asymmetries." *Psychological Review* 95 (1): 15.

Treisman, Anne M, and Garry Gelade. 1980. “A Feature-Integration Theory of Attention.” *Cognitive Psychology* 12 (1): 97–136.

Tucker, Mike, and Rob Ellis. 2004. “Action Priming by Briefly Presented Objects.” *Acta Psychologica* 116 (2): 185–203.

Tversky, Amos. 1972. “Elimination by Aspects: A Theory of Choice.” *Psychological Review* 79 (4): 281.

Tversky, Barbara, Sol Kugelmass, and Atalia Winter. 1991. “Cross-Cultural and Developmental Trends in Graphic Productions.” *Cognitive Psychology* 23 (4): 515–57.

Valdes-Perez, Raul E. 1986. “Spatio-Temporal Reasoning and Linear Inequalities.” http://dspace.mit.edu/handle/1721.1/6440.

Valentine, Stephanie, Francisco Vides, George Lucchese, David Turner, Hong-hoe Kim, Wenzhe Li, Julie Linsey, and Tracy Hammond. 2012. “Mechanix: A Sketch-Based Tutoring System for Statics Courses.” In *The Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence, Toronto, Ontario, Canada*.

Van Beek, Peter, and Robin Cohen. 1989. *Approximation Algorithms for Temporal Reasoning*. University of Waterloo, Faculty of Mathematics. https://wwwtest.cs.uwaterloo.ca/research/tr/1989/CS-89-12.pdf.

Vandenberg, Steven G, and Allan R Kuse. 1978. “Mental Rotations, a Group Test of Three-Dimensional Spatial Visualization.” *Perceptual and Motor Skills* 47 (2): 599–604.

Vilain, Marc, Peter van Beek, and Henry Kautz. 1989. “Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report.” In *Readings in Qualitative Reasoning About Physical Systems*, edited by Johan de Kleer and Dan Weld. Los Altos, CA: Morgan Kaufmann.

Voyer, Daniel, Susan Voyer, and M Philip Bryden. 1995. “Magnitude of Sex Differences in Spatial Abilities: A Meta-Analysis and Consideration of Critical Variables.” *Psychological Bulletin* 117 (2): 250.

Ward, Shawn L, Nora Newcombe, and Willis F Overton. 1986. “Turn Left at the Church, or Three Miles North a Study of Direction Giving and Sex Differences.” *Environment and Behavior* 18 (2): 192–213.

Weber, B. G., M. Mateas, and A. Jhala. 2011. “Human Level AI for Real-Time Strategy Games.” In *AAAI Fall Symposium on Advances in Cognitive Systems*.

Wetzel, Baylor, Kyle Anderson, Wilma Koutstaal, and Maria L Gini. 2012. “If Not Now, Where? Time and Space Equivalency in Strategy Games.” In *AIIDE*.

Williams, Brian C., and Johan de Kleer. 1991. “Qualitative Reasoning about Physical Systems: A Return to Roots.” *Artif. Intell.* 51 (1-3): 1–9. http://dx.doi.org/10.1016/0004-3702(91)90106-T.

Wintermute, Samuel, and John E Laird. 2008. “Bimodal Spatial Reasoning with Continuous Motion.” In *AAAI*, 1331–37.

Wolfe, Jeremy M. 1994. “Guided Search 2.0 a Revised Model of Visual Search.” *Psychonomic Bulletin & Review* 1 (2): 202–38.

Yannakakis, Georgios N., and Julian Togelius. 2011. “Experience-Driven Procedural Content Generation.” *IEEE Trans. on Affective Computing* 2 (3): 147–61.

Zhu, Liqi, and Gerd Gigerenzer. 2006. “Children Can Solve Bayesian Problems: The Role of Representation in Mental Computation.” *Cognition* 98 (3): 287–308.

Appendix A

Glossary

Anchor, Placement

See Placement Anchor.

Area of Effect

Area of Effect towers affect every enemy within range, regardless of the number of enemies. A real world example would be a hand grenade (to an extent; hand grenade effects decrease with distance and one object can act as a shield for another; neither is true in most TD games). GopherTD does not have area of effect towers.

Buffering

When one creep is the focus of a tower's attacks, allowing other creeps to move through the tower's range unharmed, the creep being attacked is acting as a buffer to protect the other creeps. We refer to the creep holding a tower's attention while other creeps are in the tower's range as a buffering creep.

Cleanup

The role of cleanup towers is to stop leakage. Typically, the intention is for the primary offense towers to stop all creeps. Cleanup towers are used as an insurance policy in the event a creep makes it past. If there is leakage, the hope is that the leaked creeps are few in number and heavily wounded. For this reason, cleanup towers are typically significantly weaker than the primary offense towers.

Creep

Generic name for the enemy that "creeps" along the path to the map exit. The goal of a TD game is to stop the creeps from making it to the exit. Many games have more than one type of creep. Common creep types are fast, freeze-resistant, flying (can go through walls; mazing games only) and divisible (when health reaches zero, the creep splits into several weaker creeps). GopherTD has a single creep type. Subject terminology included *creep*, *enemy* and *bad guy*. In other games, the creep is often a specific type of object. Known creep types include tanks (Budapest Defenders), monsters (Gemcraft), ants (Ant Buster), computer viruses (Ghost Hacker), balloons (Bloons) and laser printers (Onslaught).

Damage Over Time

Enemies hit by a damage over time tower take some amount of initial damage and then continue to take damage until the effect expires. Real world examples include being poisoned or on fire. GopherTD does not have damage over time towers.

Damage Per Second

The amount of damage done as a function of time. A tower that does 200 points of damage per shot firing 10 times a second has a damage per second of 2,000, the same as a tower that does 4,000 points of damage per shot firing once every two seconds. If two towers have identical damage per second, it does not guarantee that they will do the same amount of damage when used. For example, an object that moves through a tower's range in 3.5 seconds would be hit 35 times by the fast tower and only once by the slow tower, resulting in 75% more real damage done (7,000 vs. 4,000).

Focused Fire

Getting all towers to attack the same target (creep) so that the target can be eliminated quickly. Given two targets, in a given time frame, the result is one eliminated and the other at full health. The alternative (and more common) method is to have each tower select a target without worrying about the other towers. Given two targets, in the same time frame, both might be reduced to half health. In theory, the amount of damage dealt by the attack towers over the full time period is the same and so focused and unfocused fire should produce the same results. In practice, the number of creeps at a given time step affects the performance of the slowing towers and target selection for towers with multiple passes through the range (especially for towers with large ranges and those being used as both primary offense and cleanup) while creeps' health affects the performance of cleanup towers.

Freeze

A freezing tower causes a creep to temporarily move slowly. In many TD games, a freezing tower literally freezes the enemies (cold, ice, etc.). In others, the effect is produced by putting mud or oil on the path. Still others don't attempt to explain the mechanism. It is common for a freezing tower's slowing ability to be related to its level (i.e., upgraded towers slow creeps more). GopherTD's freezing towers (referred to as slowing towers in this documentation to avoid the connotations of freezing) slow the same amount regardless of upgrade level.

Juggling

In mazing TD games, there must always be one valid path from the map entrance to exit. Juggling is the process of opening a second valid path in order to close the first. The goal is to have all of the creeps reach the first exit and then close it and open a new exit somewhere else, forcing the creeps to go back into the maze.

Kill Zone

As creeps move through the map, they might move through areas that have no towers, areas where they are slowed, areas where light attacks "soften" targets, areas where the primary attacks take place and areas meant to catch severely weakened creeps. The kill zone is the area where the primary attacks take place.

Leak

See Leakage.

Leakage

The creeps that make it past the primary kill zones. Unless there are cleanup towers, leaked creeps will escape.

Line

Refers to a line of creeps. GopherTD has two lines, the line of green creeps and the blue line. The word group is avoided because it could refer to a line, a subset of creeps in a line or creeps in both lines.

Mazing

A TD game containing a single blank map. Creeps take the shortest path to the exit. The user can place towers anywhere and uses the placement to form mazes to control the path taken by the creeps. Mazes must have one valid path from the map entrance to exit.

Non-Mazing

A TD game that provides one or more maps containing walls. The player can only place towers on walls. Creeps must follow a predetermined path (set by the map designer) through the map.

Offense Tower

A tower whose primary purpose is to damage creeps. In GopherTD, these are the green (fast attack), red (powerful attack) and purple (powerful + slowing) towers.

Overage

The amount of damage in a shot wasted because it was more than the creep's health. For example, if a shot does 1,010 points of damage and a creep has 10 points of health, 10 points of the shot's damage will destroy the creep and the remaining 1,000 points (99%) will be wasted.

Path Synchronization Gap

When one line of creeps pulls ahead of the other, we refer to the distance the leading group is ahead as the path synchronization gap. We currently measure this in numbers of tiles. On most maps, when going around a corner, the path followed by the outside line of creeps is three tiles longer, meaning the inside line of creeps moves ahead of the outside line by three tiles.

Placement Anchor

Towers are placed on a map at a position that is relative to something. We call this something a placement anchor. The anchor could be another tower, the start of the path, the center of the map, where the usable range is highest, etc.

Placement Decision

The mental representation of a tower's location on a map. This representation is based on the intention of the problem solver. A placement decision is a triad of [placement target, placement relationship, placement anchor]. Example: In order to slow creeps as soon as they come into the range of a heavy tower, the placement decision would be [range entry, touching, range entry].

Placement Relationship

A placement relationship describes where a tower is placed, using the same features as a human problem solver. For example, it is common for a person to say that they wanted to place their offense towers on a U-turn and their slowing towers around the offense tower but no research subject stated that they were placing towers at position (8,12).

Placement Target

An item the problem solver is placing on the map. Physically, the problem solver is always placing a tower, but in the problem solver's mind they might be focusing on something else. For example, in order to attack creeps as soon as they enter the map, a problem solver might attempt to place a tower's range boundary right after the entrance to the map. The placement target in this case is the tower's range boundary, not the tower.

Rate of Fire

How often a tower fires. This is normally measured in shots per second.

Relationship, Placement

See Placement Relationship.

.

Slowing Tower

A tower whose primary purpose is to slow or immobilize creeps. In GopherTD, these are the blue (group slowing) and brown (freezing in place) towers.

Solution

In this paper we use the term solution to refer to the configuration of towers a subject has decided will give the best performance on a given map. It does not imply that the solution will guarantee a perfect score or the best score possible. Bad solutions will score lower than good solutions.

Sticky Targeting

GopherTD uses sticky targeting for all of its towers. With sticky targeting, once the tower has chosen a creep to attack, it will continue to focus its attacks on that creep until the creep moves out of range or is destroyed.

Subject

One of the people who participated in the GopherTD study. Subject refers to those people from whom we have collected empirical data . When we refer to a hypothetical person, we use the term problem solver.

Target, Placement

See Placement Anchor.

Targeting Strategy

In some TD games, the user can tell it how to select a target. Options might include closest, healthiest, weakest or user-selected. A common second option is persistence - does a tower switch targets when the criterion changes (e.g., a different creep is closer or weaker) or when the target goes out of range.

TD

See Tower Defense.

Tile

One of the grid units on the map. Maps are 18x22 tiles. Towers must align to a tile, creeps do not, moving continuously through space. Tower ranges in GopherTD (but not necessarily other TD games) are also continuous and include partial tiles. Subject terminology included *tiles*, *squares*, *cells* and *grid cells*.

Tower Defense

A puzzle in which guard towers must be placed at strategic positions to prevent enemies from reaching a specific point on a map. Guard towers attack enemies in range. The player does not directly engage in combat. Their involvement is limited to selecting locations for the guard towers.

Tower Defense Game

Each tower defense game has its own set of maps, tower types, creeps and rules. While certain tower types are common in most tower defense games, the power, cost and range of the towers are normally unique to a specific game.

Tower Defense Map

In a given Tower Defense Game there will be one or more maps. The goal in a tower defense game is to find a set of towers and tower placements that solve a specific map. The types of towers and rules for play are normally fixed across a tower defense game but the maps and the optimal solution(s) for each map vary.

Tower Defense Problem

Synonymous with Tower Defense Map. The goal of a Tower Defense Game is to find a solution to each map, meaning that each map is a problem to solve.

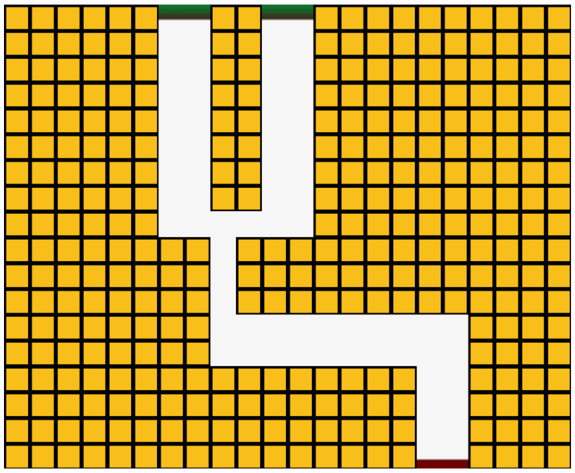
Wave

In most TD games, once a set of creeps enters the map, a timer counts down until the next set of creeps arrives. Each set is called a wave. In most TD games, the player wins when they have defended against a specified number of waves. The version of GopherTD used in the first three phases of the experiments reported here used a single wave.

Appendix B

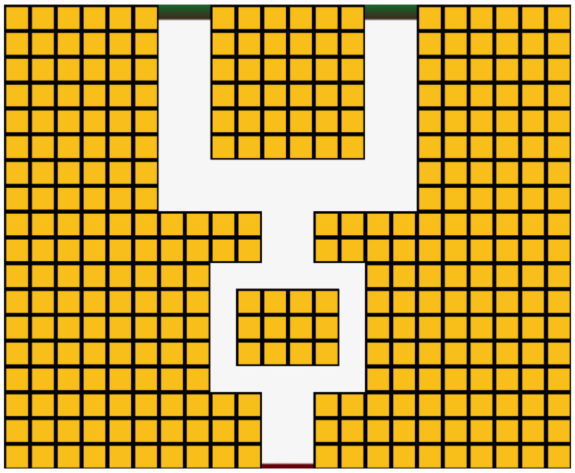
Maps

These are the 16 maps available in GopherTD. Twelve of these were used in Experiment 1, all 16 were used in Experiment 2. Maps are listed in alphabetical order. Experiments 1 and 2 were conducted under different conditions and are not directly comparable. Reported scores are for all participants tested in Experiment 2. The goal is to prevent the 28 creeps from reaching the map exit. Scores indicate the number of creeps stopped, 0-28.



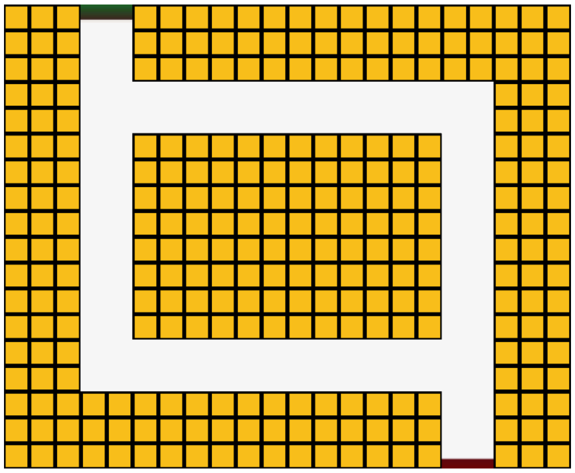
2in1out

Mean Score:	17.95
Min / Max:	14 – 23
Stdev:	2.54



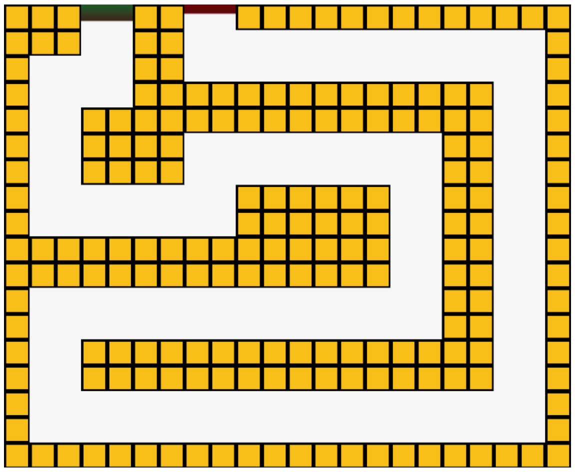
Bottleneck

Mean Score:	20.35
Min / Max:	17 - 26
Stdev:	2.06



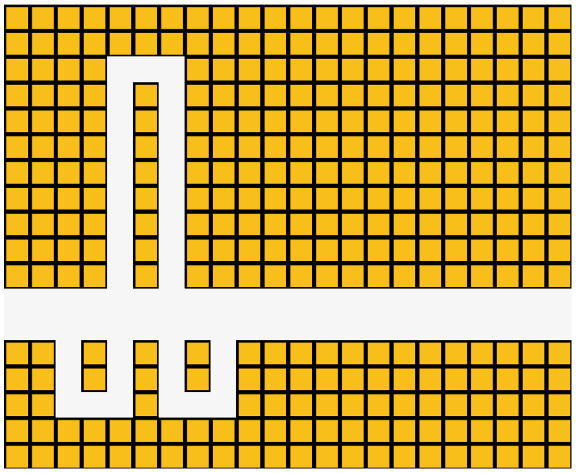
Do the Splits

Mean Score:	13.90
Min / Max:	8 – 21
Stdev:	3.48



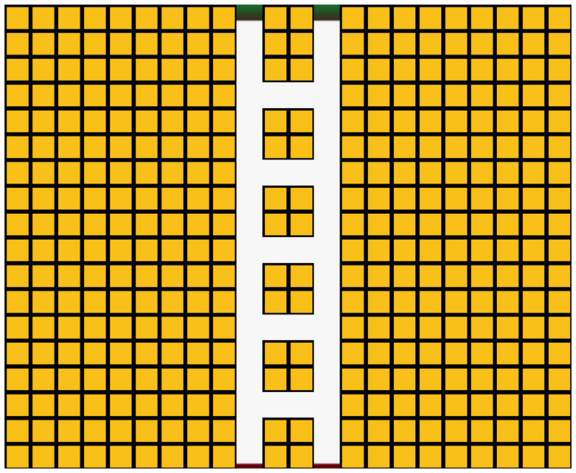
Elemental-ish

Mean Score:	27.20
Min / Max:	21 - 28
Stdev:	1.64



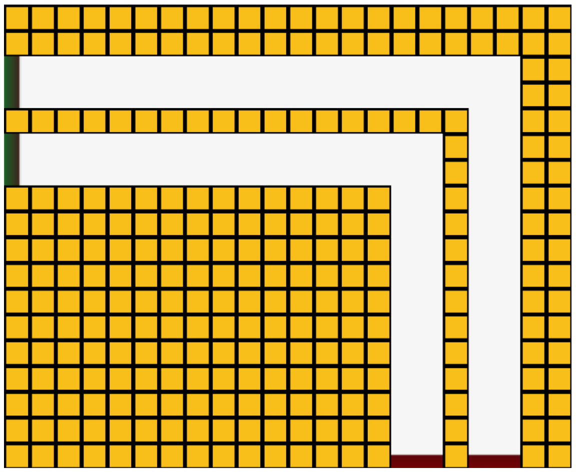
Finger

Mean Score:	26.15
Min / Max:	22 - 28
Stdev:	1.90



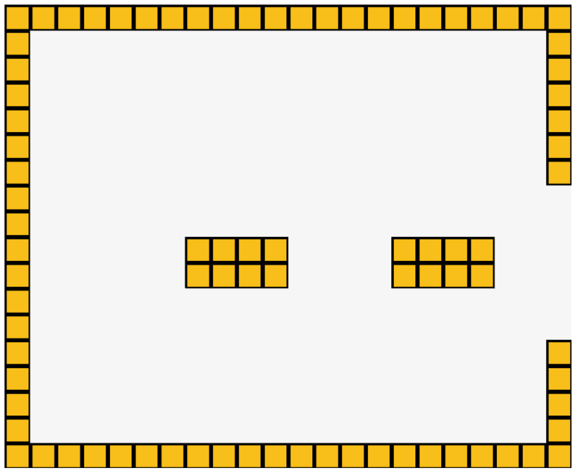
Ladder

Mean Score:	23.7
Min / Max:	20 - 27
Stdev:	2.08



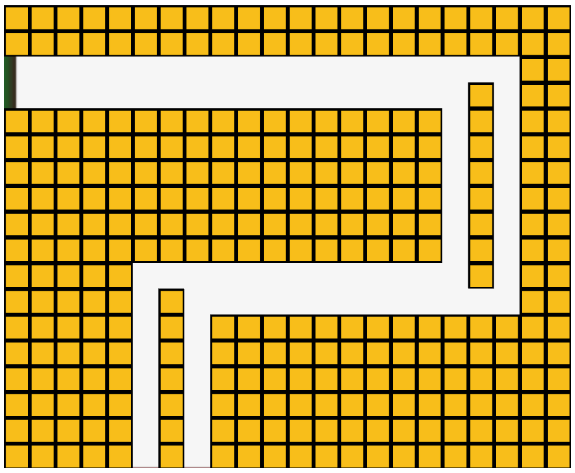
No Left Turns

Mean Score:	15.55
Min / Max:	11 - 28
Stdev:	4.50

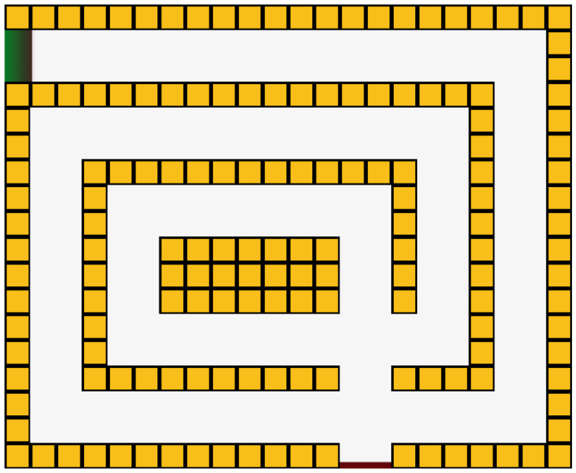


No Support

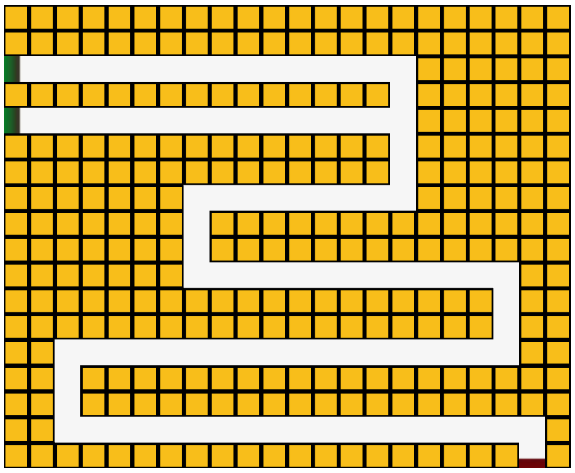
Mean Score:	25.65
Min / Max:	17 - 28
Stdev:	2.54



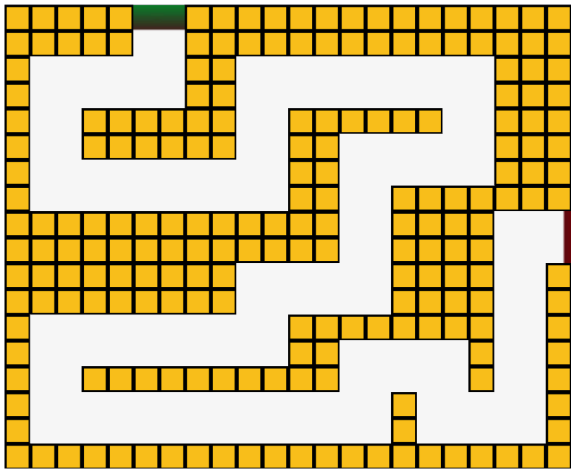
Pathways
Mean Score: 19.10
Min / Max: 10 - 28
Stdev: 5.92



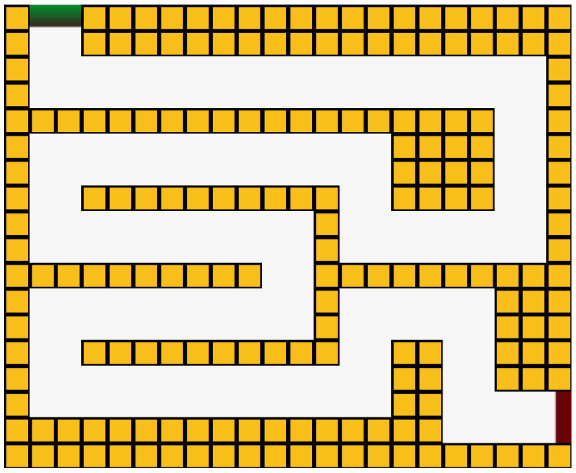
Round the Twist
Mean Score: 28
Min / Max: 28 – 28
Stdev: 0.00



Slim Pickings
Mean Score: 27.10
Min / Max: 17 - 28
Stdev: 2.49

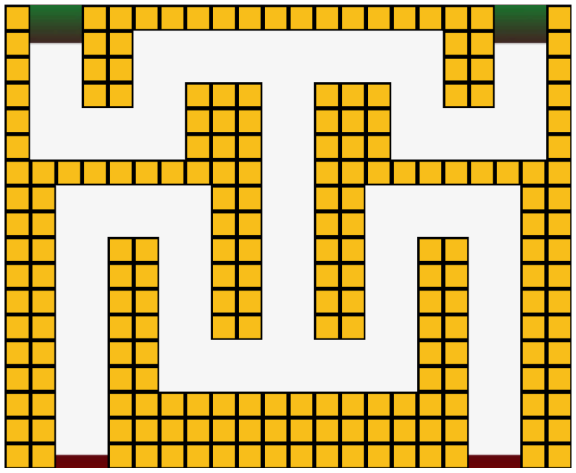


Snaking Path
Mean Score: 27.50
Min / Max: 26 - 28
Stdev: 0.69



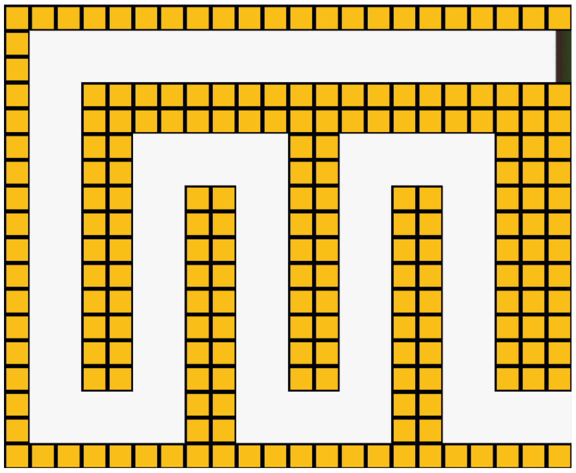
Switchback

Mean Score:	28
Min / Max:	28 - 28
Stdev:	0.00



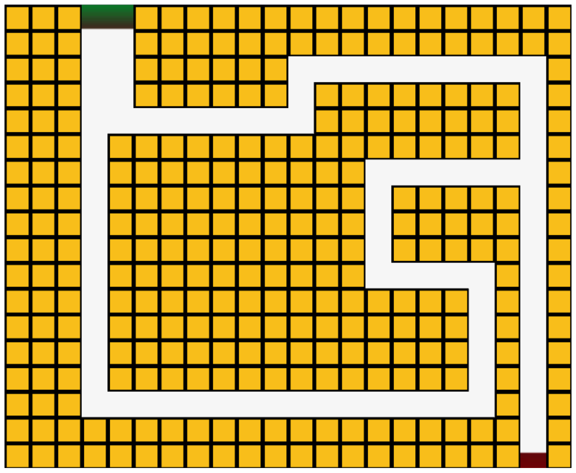
The Frog

Mean Score:	25.25
Min / Max:	20 - 28
Stdev:	2.63



Up N Down

Mean Score:	26.75
Min / Max:	15 - 28
Stdev:	2.88



Void

Mean Score:	22.80
Min / Max:	18 - 28
Stdev:	3.25

Appendix C

Methodology and Issues in Human Experiments

To study how humans solve complex problems with significant spatio-temporal components, we collected data from 58 people over two experiments. All experiments required subjects to solve a set of tower defense problems in GopherTD under a think aloud protocol. Experiment 1 (n=38) looked at learning from experience, transfer performance, methods of reasoning and representations used. In Experiment 2 (n=20), subjects without any tower defense experience were compared to those with prior experience on other tower defense puzzles but not on GopherTD to determine how expertise contributed to strategy selection and performance.

Experiment time ran from 90 minutes to four hours. The majority of sessions in Experiment 1 took 2-3 hours. The majority of sessions in Experiment 2 took two hours. Aside from short bathroom breaks, no breaks were taken during an experiment.

Objective data (scores, solutions, etc.) were automatically collected by the experimental tool, GopherTD. Subjective data (intentions, mental representations, etc.) were collected during the task using a think aloud protocol and after the task in an after-action review.

Subjective data were analyzed by first fitting it to a paper behavioral model. This model was iteratively developed – two coders used the model to encode subject data and in the event of significant inter-rater disagreement or data that could not be properly described, the model was changed to better explain the data. Changes were also made if it was determined that a simpler encoding could be used to replace a more complex one.

To validate our understanding of subject behavior, we took the paper behavioral model and implemented it as a computer model. This process uncovered areas where assumptions had been made and detail was lacking. The computer model was a generative model, meaning it not only described a subject's behavior but could generate solutions in the subject's problem solving style. Solutions created by the artificial agent were compared to the subject's to verify that the proper data had been captured and that it accurately described the subject's behavior. We used a single agent to replicate all subjects' behavior to ensure that the model we used and mechanisms we designed were sufficiently general.

C.1 Protocol

Two experiments were conducted. Each experiment had a training phase, a test phase and an interview phase. The task in both experiments involved solving tower defense problems in GopherTD.

Data was collected in several ways.

During testing and training, subjects solved puzzles in GopherTD. In training, only the number of solution attempts and time spent on each is collected.

In testing, GopherTD tracked data related to process (time spent planning, towers used, order they were placed) and performance (score, tower activity). During the testing phase, subjects solved puzzles under a think aloud protocol, saying everything that came to mind. During testing, the computer recorded audio and video, recording what was seen on the screen (images and mouse movements) and what the subject said out loud. The experimenter took notes during this time, paying special attention to the subject's use of terms, goals and intention. Unclear material was flagged for clarification during the after-action review.

GopherTD logged all of the subject's solutions. After the testing phase, the subject was shown their solution to each problem and asked to explain it. The interviewer asked about goals and intentions. Many questions were compare/contrast – if the subject stated they were following a certain strategy and multiple positions matched their description, they were asked why they chose the location they used and not the others. As with testing, the screen and the subject's speech were recorded.

After the experiment, the recordings were rendered to video and archived. Each of the subject's solutions (including intention, goals and representations) were encoded (on the paper behavioral model) by two coders.

C.1.1 Equipment and Facilities

Subject data was recorded by GopherTD, a full-screen application created for this research. Camtasia 1.2.0 was used to record screen information (screen images and mouse movements) and spoken information from the experimenter and subject.

Experiments were conducted on a MacBook 5,2 running Mac OS X 10.5.7. Resolution was 1280x800 on a 13.3" glossy widescreen LCD. Video was hardware accelerated by an NVIDIA GeForce 9400M. There were no reported problems with rendering (i.e., no lag, artifacts).

Experiments were conducted in a quiet room. The only two people in the room were the subject and experimenter.

C.1.2 Automated Performance Capture

GopherTD records the following information for each problem a subject solves:

- **ID**. The assigned ID of the subject. The ID includes the experiment number, subject category (domain novice or experienced) and a sequential number.
- **Date**. Date the problem was solved.
- **Problem Number**. In the testing phase, subjects are presented with a sequence of problems (maps) to solve. The problem number is a sequential number indicating which problem they are on.
- **Map Name**. Name of the map. For a given experiment, all subjects are presented maps in the same order. This order was not preserved across experiments so map name is necessary to identify which problem was solved.
- **Score**. The number of creeps the subject stopped.
- **Remaining Funds**. The amount of money the subject did not spend to purchase towers. We know of no good reason to not spend all available money but captured this to be sure. Barring mistakes, subjects always spent all available funds.
- **Build Time**. Time spent selecting and placing towers. Represents the amount of time a subject spent to solve the problem.
- **Wave Time**. The amount of time spent watching the creeps move through the map. The subject can control the speed the creeps move at. This data tracks a mixture of how long it takes the creeps to move through the map (which is influenced by the path length and use of slowing towers) and how much attention

the subject spends watching (and presumably studying the execution of) their solution.

- **Tower Statistics**. Information about each tower.

Information tracked for each tower includes:

- **Type**. The type of tower (Blue1, Green3, etc.).
- **Level**. The level (1-10) of the tower.
- **Position**. (x,y) position of the tower. This information is used by the solution saver, log viewer and replay systems to re-load solutions for modification, viewing and replaying respectively.
- **Placement Number**. Records the order the towers were placed in. This is used to study placement information such as whether subjects consistently place offense or slowing towers first and whether subjects place towers in spatially proximate groups first or distribute them. Only towers that are used in the final solution are tracked (i.e., towers that are placed then later removed are not included) and the data cannot distinguish between towers that are removed because they are no longer needed and towers that were removed so that they could be moved to a new location.
- **Total Damage**. How much damage the tower did to creeps.
- **Total Excess Damage**. How much damage was done that was not necessary, known as *overage*. This concept is discussed in Section 8.4.3 To Reduce Overage.
- **Average Excess**. Mean overage (total overage / number of overage shots).
- **Number of Shots**. How many times the tower fired.
- **Number of Overage Shots**. How many shots had overage.

In addition to this information, GopherTD saved screenshots of each solution. These were used during coding to visualize subject actions.

Analysis and visualization of some data was done using additional applications created by the experimenters. As an example, the tool GTDLogFileAnalyzer was created to determine how consistent subjects were in their choice of towers.

C.1.3 Think Aloud Protocol

The testing phase is conducted under a think aloud protocol (Lewis 1982). Subjects must say out loud every thought that is going through their head, without filtering, even if the thought is not directly related to the task at hand. Subjects are asked to not talk to the experimenter or explain what they're doing as this causes subjects to remove themselves from their thought process and consciously deliberate on what they're doing. Self-reflection can also modify a subject's behavior. Attempting to explain one's behavior causes them to think about whether their ideas sound good and revise them if they do not. It can also lead to subjects giving an explanation that does not match what they are doing. This problem occurred frequently in the after-action reviews so it was important that the information recorded during the testing phase be unedited, stream of consciousness information.

When working properly, the think aloud protocol does not interfere with the subject's task performance or interrupt their flow. Although acting while thinking aloud can feel awkward at first, the self consciousness tends to disappear after a few minutes. Although we did not measure it, we believe that roughly half of the subjects had no problem thinking aloud for the entire experiment.

For the other half, thinking aloud did not appear to affect their thoughts but there were two areas where they were likely to stop speaking (subjects must speak continuously; if they stop, the experimenter prompts them with the phrase "keep talking"). The first area was when the subject was thinking deeply. This typically happened when the subject was considering their overall plan. A typical dialog is "Okay, new map, there aren't a lot of corners here but I think I can still use my strategy, let's see...", at which point they stop talking.

The second area involved the situation where the course of action had been determined and the actions to take were repetitive and required little thought. The most common example of this involved the SLOW IN RANGE strategy. In this strategy, subjects place their offense towers, then surround each offense tower with a number of slowing towers. A typical statement was:

> "Okay, the main towers are placed, now I want to make sure the creeps move slowly through these areas so I'm going to surround each tower with a bunch of slowing towers"

after which they placed a large number of towers (in Experiment 2, 20 blue towers) one by one around the offense towers. There is a tendency for subjects to stop talking after placing the first few towers, with some subjects arguing that there literally is nothing to say because they are no longer thinking, they are simply placing towers on auto-pilot.

All think aloud data is recorded. The experimenter also takes notes. Originally this meant filling out the coding sheet while the subject talked but this proved too time consuming so we switched to taking free form notes. In the subject's speech we listened for several things:

- **Terminology**. We recorded the terms subjects used, predominantly representational terms. There was little consistency in this terminology.
    - To describe the area the creeps moved through we heard the terms *corridor*, *column*, *path*, *pathways*, *lanes*, *firing lanes*, *tracks*, *thoroughfares*, *channels*, *legs*, *room*, *ring*, *quadrants*, *ground*, *sides*, *stretch*, *places*, *squares*, *maroon* (the color of the path area), *maroon squares* and *purple path*.
    - Maps had *corners*, *turns*, *bends*, *interior corners*, *loopbacks*, *switchbacks*, *zigzags*, *areas*, *blocks*, *islands*, *partial islands*, *360s*, *circles*, *bottlenecks*,

U-turns, *horseshoes*, *folds*, *curves*, *pegs*, *spikes*, *columns*, *rows*, *bars*, *strips*, *peninsulas*, *thin walls*, *passes*, *intersections*, *triple junctions*, *spawn points* (a common term in other game genres), *entry points*, *exits*, *finish lines* and *things*.

  - To separate one line of creeps from another, subjects said they were *spreading*, *separating*, *spacing*, *pulling and selectively slowing* the enemies, referred to as *creeps*, *lines*, *strings*, *strands*, *waves*, *rows*, *enemies*, *pieces*, *herd*, *guys*, *groups*, *barbarians*, *mobs*, *people* and *little guys*.

  In the end, the benefit for collecting this data (terminology, not representations) was small but it did help identify representations and strategies we might not have otherwise noticed, such as SATISFICING STRATEGY: COLOR RATIO.

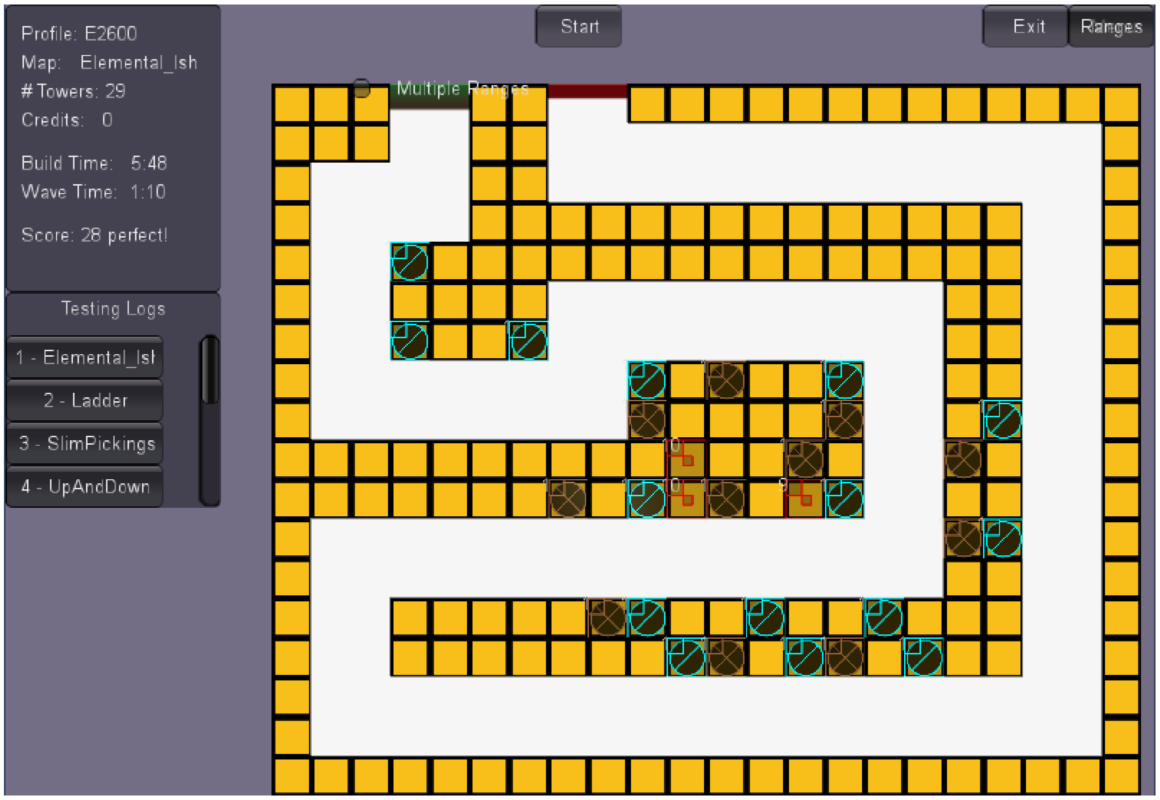
- **Goals, Intentions and Strategies**. GopherTD is able to capture the moves the subject makes but not the reasons for them. It was (and continues to be) our belief that the key to transfer learning is generalizing the reasoning for actions rather than the actions or solutions themselves. We paid careful attention to the reasons subjects gave for their decisions. While we gained much valuable information about strategies from the think aloud data, this information often needed clarifications that could only be achieved in the after-action review.
- **Problems**. As described in Section 5.1 Solution Analyzed: The Problem Solving Approach of E1500, subjects appeared to learn strategies by noting a small, simple problem, stating the impact and importance of the problem and immediately determining a solution to the problem. Much of a subject's problem solving style appears to be based on noting problems, setting goals and determining actions based on those. While not all solutions worked, understanding which problems motivated them helped us understand the nature of problem solving.

In addition to the above, the experimenter noted items that needed clarification so that they could ask the subject about them in the after-action review.

The think aloud data was extremely valuable in understanding how people thought about the problems. In many instances, what was said during think aloud and what subjects later claimed they were thinking did not match. In these cases it seemed the think aloud data was more accurate.

C.1.4 After-Action Review

After the testing phase was completed, subjects were shown their solutions and asked to explain their reasoning. Because the experimenter is not allowed to talk during the testing and think aloud portion, the after-action review was the first chance for the experimenter to ask questions about the subject's intentions and reasoning. The after-action review was audio recorded.


Profile: E2600
Map: Elemental_Ish
Towers: 29
Credits: 0

Build Time: 5:48
Wave Time: 1:10

Score: 28 perfect!

Testing Logs

1 - Elemental_Ist
2 - Ladder
3 - SlimPickings
4 - UpAndDown

Figure C.1. GopherTD Log Viewer. GopherTD's log viewer shows the solution subject e2600 created for the map Elemental-ish. The log viewer shows how long the subject spent building it and what score it earned.

The after-action review used GopherTD's log viewer (Figure C.1). The experimenter selects the subject and then the map to look at. The log viewer shows the map, the subject's solution, the amount of time they spent creating it and the resulting score. The log viewer allows the subject and experimenter to select a tower, showing the tower's range and area it covered, which is often important for explaining why a tower was placed where it was. Multiple towers can be selected so that the relationship between towers can be explained.

The log viewer was designed to makes it easy to switch back and forth between maps. This is useful for investigating how a strategy is applied across problems and how it evolves over time.

One of the problems with spatio-temporal problems is that they are hard to explain or analyze from a static picture. The log viewer has the ability to replay a solution. This allows the subject to show rather than describe their ideas.

The replay feature was also used heavily during analysis to understand when and why certain strategies did or did not work. Because it can show solutions generated by AI agents, this tool was also used to study, debug and test changes to the better-than-human computer agents.

C.2 Demographics

C.2.1 Categories

The study consisted of 58 subjects divided into four categories: Domain Experienced, Domain Novice, Author and Special. The latter two categories reflect the creators of GopherTD and people who had heard talks on GopherTD strategies before playing. The majority of subjects were in the first two groups.

- **Domain Novice**. Have never heard of or solved a tower defense puzzle. Includes a few subjects who had spent less than 15 minutes with tower defense.
- **Domain Experienced**. Have played one or more tower defense games extensively, usually for several years. Understand the concept (stop creeps by placing towers) and the purposes of different types of towers. Domain experienced subjects understood how the domain worked but were not necessarily good at it. Subjects' self-evaluation of skill ranged from “okay” to “expert” and scores showed a large spread in ability.
- **Author**. GopherTD was created by two programmers. Both authors had extensive experience with Vector TD, the game GopherTD is based on. Both subjects had more complex strategies than demonstrated by other subjects, including map-

specific tower selection strategies, which no other subjects demonstrated. They did not, however, have the highest scores in this study.

- **Special**. Subjects in this category were people (three domain novices, one domain experienced) who had heard talks on the GopherTD study prior to participating in the experiments. The subjects did not appear to use the strategies mentioned in the talks. It is not known why this is.

	Experiment 1			
Category	**Phase 1**	**Phase 2**	**Experiment 2**	**Total**
Experienced	12	17	10	39
Novice	1	2	10	13
Author	1	2	-	2
Special	-	4	-	4
Total	14	25	20	58

Table C.1. Number of Subjects by Experiment and Experience.

It is important to note that these categories are domain novice and experienced, not domain novice and expert. Domain experienced subjects have solved tower defense puzzles before. While they have not solved GopherTD problems before, they are aware of how tower defense puzzles work in general, lowering the amount of training they need. Having domain experience did not guarantee that the subject was good at solving these puzzles.

Because domain experienced subjects were presumed faster at training, we strove to use domain experienced subjects where possible in the first two experiments. In Experiment 2 we used an equal number of domain novice and experienced subjects as we wanted to look at the effects of domain experience.

C.2.2 Background

Subjects were primarily undergraduate engineering and computer science students from the University of Minnesota. Because this was the initial population subjects came from, we stayed with this population in later experiments to control for educational focus as a variable. Engineers came from the departments of biological, chemical, computer, electrical and mechanical engineering. It is theorized that engineering students reason about space differently than students in spatial majors (e.g., art, architecture) and non-engineering majors (e.g., psychology). This is a possible direction for future work.

Variable	Min	Mean	Max
Age	18	22.34	44
Years of Education	14	15.86	21

Variable		
Gender	46 M	12 F
Handedness	46 R	5 L

Table C.2. Demographic Information for Study Participants. 14 years of education represents a college freshman. Counts for handedness is lower than for gender as some subjects declined to fill out all fields on the demographic survey.

Gender was not evenly split between categories. Finding male engineers who had played tower defense games was not a problem but it was difficult to find women with domain experience. Previous research has shown that gender plays a strong role in spatial reasoning. Given the population we used, there is no way to tell if there was a gender effect in this research.

Category	Male	Female	Percent Female
Novice	6	9	60%
Experienced	39	3	7%

Table C.3. Gender of Subjects. Gender and experience were strongly correlated.

Experiment 1 used primarily domain experienced subjects. The majority (67%) of domain novices participated in Experiment 2. The majority of domain novices recruited for Experiment 2 came from an introductory programming course for honors students. All were either engineering or physics students. It is unknown whether having only one group drawn from honors students impacted the results, although if it did it presumably biased their scores upwards and the honors students, who did not have domain experience, scored lower than non-honors students with domain experience. We therefore believe that the novice vs. experienced gap reported in Experiment 2 is, if anything, understated.

C.3 Changes between Experiments

In order to achieve our goal of building computer agents that can solve complex spatio-temporal problems we collected data on how humans solve these problems. Detailed data existed for how people solved simple problems and conceptual data existed for how experts solve complex problems but we were unable to find detailed explanations for how people solved complex problems. For this reason, it was unclear how to begin this research. The goal of the first experiment was to obtain broad, general knowledge that would allow us to focus our investigation.

Experiment 1 was conducted in two phases. The first phase helped us work out methodological issues. The need for this became apparent running the first subject, a process that took eight hours. This was partially due to the subject being particularly meticulous and partially due to the subject having no previous experience with tower defense. The length of the experiment required the subject, a student, to take several breaks to attend classes. When the subject returned from the breaks, they had forgotten or misremembered important details, which contributed to the length of the experiment and negatively impacted performance. Despite these problems, we learned a significant amount about problem solving and representations, but this experiment led to our restricting the remainder of Experiment 1 to people with tower defense experience. Subsequent experiments took 2-4 hours (mode 3), which proved to be an acceptable amount of time.

There were eight maps in Experiment 1 Phase 1. Seeing the 7th map (Void) caused many subjects to create a new strategy which they then used on maps 7 and 8. Although it was fascinating to see that simply viewing a problem could cause an insight that led subjects to develop an advanced temporal-focused strategy, it ran counter to our purpose of studying subjects after they had reached response stabilization. In Experiment 1 Phase 2, this map was made a training map.

	Experiment 1			
Category	**Phase 1**	**Phase 2**	**Experiment 2**	**Total**
Experienced	12	17	10	39
Novice	1	2	10	13
Author	-	2	-	2
Special	-	4	-	4
Total	13	25	20	58

Table C.4. Number of Subjects by Experiment and Experience.

GopherTD contains 11 types of towers, each of which has 10 power levels, resulting in 110 options for each tower placed on the map. Subjects used sets of towers in any combination they chose. Towers vary in cost, affecting the number of towers a problem solver can use. Using the cheapest towers, a problem solver can place 500 towers while using the most expensive towers limits the subject to just three.

Experiment 1 Phase 1 gave subjects access to all 11 towers. Although many different types of towers were used in successful solutions, a review of Phase 1 data showed that certain towers were strongly correlated with low scores. In Phase 2, the number of available tower types was reduced from 11 to 4, reducing the number of options from 110 to 40. This reduced the range and standard deviation of scores in Phase 2. For Experiment 2, where our interest was on measuring the performance of placement strategies, we removed the influence of tower selection strategies by requiring all subjects to use the same set of towers. The set of mandated towers was determined by reviewing the towers used in the highest scoring solutions and running 1,000 iterations of a simple strategy test using each tower and mixture of towers. The impact of pruning towers that were associated with lower performance on scores can be seen in Table C.5.

	Experiment 1 Phase 1 Score Averages				Experiment 1 Phase 2 Score Averages				Experiment 2 Score Averages			
Category	**min**	**mean**	**max**	**std ev**	**min**	**mean**	**max**	**std ev**	**min**	**mean**	**max**	**std ev**
Experienced	6	19.57	27	7	14	21.77	27	4	21	24.21	27	2
Novice	21	21.00	21	0	11	13.17	15	2	19	22.66	25	2
Author	-	-	-	-	24	25.13	26	1	-	-	-	-
Special	-	-	-	-	13	18.60	26	6	-	-	-	-

Table C.5. Scores By Experiment and Experience. Average minimum, mean and maximum scores for each experiment. Scores are not directly comparable across experiments as changes were made in available maps and towers.

We believed that how subjects chose towers would be an interesting topic. Different towers have different capabilities and afford different strategies. In our analysis, between-subject variance for tower selection was high but within-subject variance was not. It was not clear from our data why subjects chose the towers they did but once subjects selected a set of towers they stuck with them. This was disappointing. Advanced Vector TD players often select their towers based on the properties of the map but we did not see this in our subjects. Subjects were loyal to a set of towers regardless of how poorly they were doing with them. This observation supported our decision to assign subjects a set of towers in Experiment 2. We suspect that the amount of time provided for training was not sufficient for subjects to develop a robust set of tower selection strategies and hypothesize that if we had chosen subjects with considerable experience with Vector TD or GopherTD we would have seen more within-subject variance.

Experiment 1 produced a tremendous amount of data on how people represent and solve complex spatio-temporal problems. Having gathered data on breadth, our attention turned to performance. In Experiment 2, all subjects were assigned a set of towers, removing tower selection strategies as a variable in subject performance. Because many strategies are dependent on affordances only offered by certain towers, removing towers pared down the number of potential strategies. Because our interest was in how well specific strategies worked rather than gathering breadth data or measuring transfer performance as in the first experiment, all maps were available in both testing and training.

Experiment 2 looked at performance as a function of the specific strategies used. We also wanted to examine the impact of having domain experience. Experiment 2 was the only experiment where we actively recruited subjects with no tower defense experience. Because training was limited to three attempts per map (48 trials over 16 maps), domain novice and experienced subjects took roughly the same amount of time to complete the experiment (range of 90 minutes to 3 hours, mode of about 2 hours).

C.4 Approach to Data Analysis

A prominent feature of this work is the use of computer models to validate, clarify and uncover issues with human data. Computer science was essential to the psychological contributions of this work. In addition to the psychological contributions of this work, a goal of this work is to create human-level or better-than-human spatio-temporal problem solving agents. The psychological work here was critical to the computer science contributions of this work.

Given the interdisciplinary nature of this work, it is worth spending a little time discussing the approach this work took to data analysis.

C.4.1 Behavioral Model

Cognitive modeling focuses on accurately describing low-level mechanisms that make the brain work. These models can make accurate predictions about low-level phenomena (e.g., the power law of practice) but are often too low to provide insight into task-level behavior (i.e., solving problems). Behavioral modeling takes a more black box approach, focusing on higher-level behaviors and the concepts and representations that make them possible. Behavioral modeling makes no claims about lower-level cognition beyond what is necessary to explain the modeled behavior.

The research described here is behavioral modeling, not cognitive. We present a significant amount of data related to the data we observed and mechanisms that would allow for that behavior but we are not arguing that these descriptions or the generated computer model presented in Chapter 9 Computer Model is an accurate description of how the mind works. One of the well known problems in experimental psychology is that there is no way to directly measure what is inside a mind, we can only infer the internal processes and representations from the behavior we can measure. The only claim we make is that the model presented here is capable of explaining all of the behavior of interest at the level of behavior (task-level) that we're interested in.

C.4.2 Paper Model

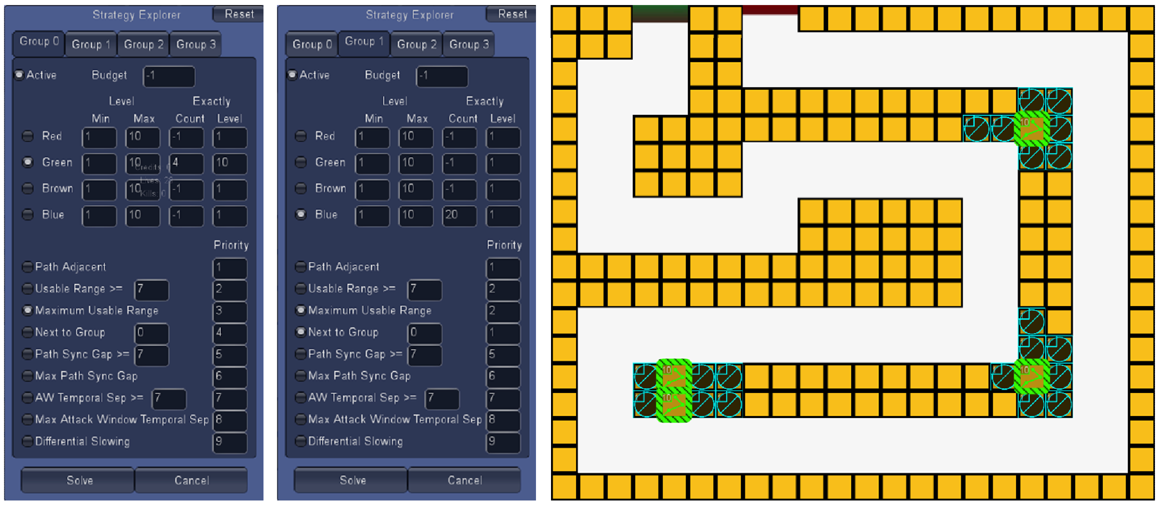
Before creating a computer model of the GopherTD data, we first created a paper model. A key challenge and contribution of this research was finding the right level of abstraction to use. As became clear during the conduct of this work, much of human intelligence relies on finding the right level of abstraction. Too concrete and the model can't help with novel situations, too abstract and the model can't exploit useful information. The GopherTD behavioral model is detailed enough to create generative models of each individual but general enough to describe the problem solving approach of all 58 subjects. This model was continually updated during the experiments, allowing us to simplify a large number of variables down to a smaller (but still large) number of key concepts, strategies and representations, creating a complex yet parsimonious description of behavior.

C.4.3 Computer Model

In experimental psychology, theories and models are often created and interpreted by humans. Creating computer models not only benefits psychology by providing an additional means of validating these theories and models, it benefits them by forcing the researcher to specify all of the relevant details. When only humans are involved, details can be glossed over because the meaning of them is considered "obvious". By forcing researchers to precisely define all terms, variables and processes, creating computer models can lead researchers to uncover ideas that are just as valuable as the original idea being studied. Some of the most important concepts in this research came from difficulties in converting a paper model we thought we understood into a computer model.

C.4.4 Generative Model

Many behavioral theories of decision making are aggregate descriptive – they describe existing data but do not make predictions or they explain overall patterns but do not predict individual actions. For example, the generalized matching law can predict how often a person will select option A over option B but not what they will do at the next choice point. This work uses generative behavioral models, which executes a given behavior rather than just describing it.



Group 0: ● Active, Budget -1

	Level Min	Level Max	Exactly Count	Exactly Level
Red	1	10	-1	1
● Green	1	10	4	10
Brown	1	10	-1	1
Blue	1	10	-1	1

		Priority
Path Adjacent		1
Usable Range >=	7	2
● Maximum Usable Range		3
Next to Group	0	4
Path Sync Gap >=	7	5
Max Path Sync Gap		6
AW Temporal Sep >=	7	7
Max Attack Window Temporal Sep		8
Differential Slowing		9

Group 1: ● Active, Budget -1

	Level Min	Level Max	Exactly Count	Exactly Level
Red	1	10	-1	1
Green	1	10	-1	1
Brown	1	10	-1	1
● Blue	1	10	20	1

		Priority
Path Adjacent		1
Usable Range >=	7	2
● Maximum Usable Range		2
● Next to Group	0	1
Path Sync Gap >=	7	5
Max Path Sync Gap		6
AW Temporal Sep >=	7	7
Max Attack Window Temporal Sep		8
Differential Slowing		9

Figure C.2. Strategy Explorer. The Strategy Explorer allows the experimenter to try different strategies. In this example, the experimenter has asked for the ADVANCED MAXIMUM USABLE RANGE strategy used by most domain novices in Experiment 2. Uses the same towers as in Experiment 2. LEFT: Asking for the offense towers to be placed where they will get the MAXIMUM USABLE RANGE. CENTER: Asking for the slowing towers to be placed near the towers in the first group and, within those positions, where they will have the MAXIMUM USABLE RANGE. RIGHT: The resulting solution.

Figure C.2 shows GopherTD's Strategy Explorer. This tool allows the user to select one or more strategies and combines them to produce a solution to a given problem. Using this tool, an experimenter can specify the strategies of a given subject and generate a solution in that subject's problem solving style. This solution can be compared to the one created by the subject to determine how accurate the computer model is. Section C.5.4 Unrecognized Vagueness discusses the situation where the original implementation of the spatial relationship "near" was objectively correct but generated incorrect behavior, leading to an insight into the context-dependent nature of spatial relationships. This issue was discovered through the use of Strategy Explorer. Without a generative computational behavioral model, it is unlikely that this issue and the insights it inspired would have been detected.

C.5 Issues and Implications

C.5.1 Task Complexity

Many subjects had difficulty in their strategies in a way that matched what they were doing.

It takes longer to learn how to solve complex problems than simple ones. Our first subject, a domain novice, spent four hours in training. The average expert spent one hour. If we decided to collect data from domain novices, we would need to commit to longer experiments. This had several disadvantages. Subjects cost more, we could run fewer of them, many could not find an uninterrupted eight-hour block of time and concentration waned, affecting performance and making it difficult to measure what we wanted to measure. We were obligated to give our subjects breaks and found that their scores plummeted, the subject having forgotten or otherwise revised their strategies over the space of a 15-minute break.

One alternative to long training sessions was to set limits on training time. However, given the complexity of the task, if we limited training time, subjects failed to reach response stability. Since subjects continued to learn during testing, we were unable to draw any conclusions about transfer learning. The desire to continue experimenting during testing was present for all subjects but was more pronounced among those who felt they had inadequate training time.

In our work we've found that subjects strongly distinguish between positional and tower selection strategies. Under normal testing conditions, most subjects chose a set of towers and used those for all problems. While this might be an interesting finding, we believe that this is more likely a result of not having enough time during training to develop both positional and piece selection strategies.

One way to reduce training time is to find people already experienced with the domain or similar problems. In our case, this meant recruiting people with experience with other tower defense games.

Experiments that rely on experts potentially have problems recruiting a sufficient number of subjects. Given the popularity of tower defense games, this was not a problem in terms of sheer numbers. However, the majority of subjects were from computer science or engineering. Only eight of the 58 subjects (14%) were female.

There are many reasons why psychologists use mice to study decision making and learning but one of the less talked about ones is that it allows the researcher to know the subject's entire learning history. While knowing what experience a subject has had is valuable to any researcher, it is especially important for those studying how people solve complex problems.

In our research, we were interested in how people acquired and applied problem solving strategies for the tower defense domain and how they transferred these strategies to other domains. If we used experts, we had people who had strategies, thus letting us study strategy application but not acquisition. We would be unable to see how they learned their problem solving strategies. If we brought in non-domain experts, the concept of transfer learning that we were interested in suggested that subjects would solve problems in our domain by transferring strategies from other domains they were experts in. Without knowing the learning history of the subject, it didn't matter whether they were a domain novice or not, they would make decisions based on things they had learned prior to the experiment. It appears to be impossible to study the complete process of how humans learn to solve complex problems.

C.5.2 Think Aloud

Think aloud data was often but not always accurate. The reasons for this are unknown. One possibility is that statements relating to beliefs are not as accurate as statements related to actions, possibly because beliefs are ephemeral, prone to change in an instant with no cognitive dissonance. An example is shown in Figure C.3. On the map Finger, the top line of creeps heads straight for the exit while the bottom line of creeps moves through a series of curves, arriving at the map exit long after the first line has exited. When placing their offensive towers, subject 3E1000 said that they were focusing on the end of the path since towers in that position would be able to attack both lines of creeps and do so in two separate attack windows (i.e., they could attack the top line of creeps and then later attack the bottom line). This is a sensible strategy and would have resulted in a solution similar to the one in Figure C.3 (left). Instead, subject 3E1000 placed the towers at the positions shown in Figure C.3 (right). The last two towers match the position the subject described but the first two do not. The towers are in positions that have a higher usable range, which is a good thing, but they do not match the placement, strategy or reasoning the subject stated in their think aloud.

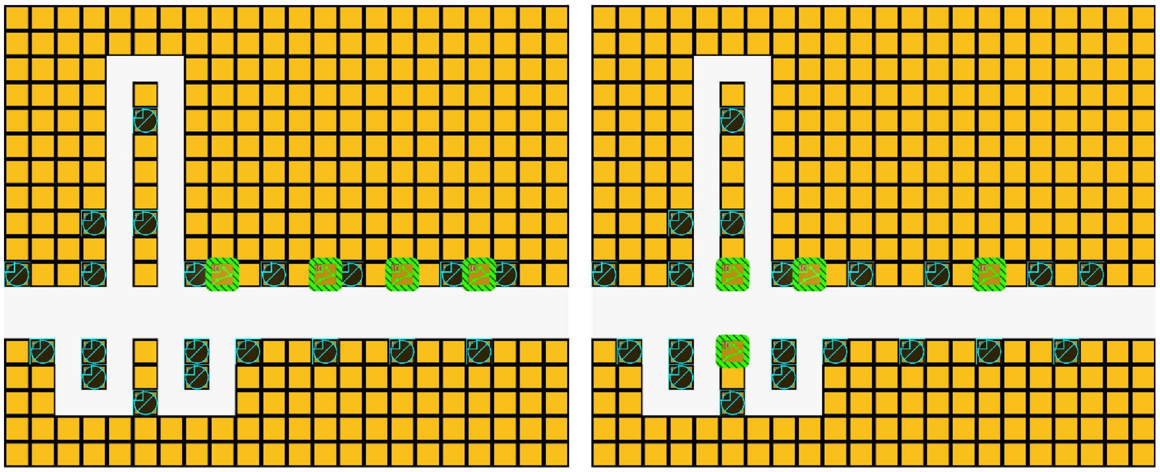


Figure C.3. Discrepancy Between Reported Behavior and Actual. Subject 3E1000 said they will place their offense towers (highlighted) at the end of the path. LEFT: Where the towers should have been placed. RIGHT: Where the towers were placed.

While we do not know why this discrepancy appeared, one possibility is that the subject intended to follow the strategy when they stated it and placed the first tower but changed their mind while placing the remaining towers and something blocked them from stating the change in strategy. Possibilities for this failure to announce the strategy change include the subject not being aware of the change (i.e., it happened subconsciously) and the change happening too fast to report (i.e., they changed their strategy instantly and moved to a new topic, which they started to talk about; under a think aloud, subjects should not stop actions or new thoughts to report old ones). Another possibility is that the action reflects a habitual "action capture" error, where the subject automatically and inadvertently placed some of the towers according to a more frequent and common placement.

Issues with the think aloud protocol are outside of the focus of this work and we have no plans to look into this in future work. For our purposes, it is sufficient that the think aloud protocol worked well the majority of the time and provided a significant amount of important information that could not have been gained any other way. We note these problems so that we are aware of them and can work around them.

C.5.3 After-Action Review

During after-action review it was common for subjects to forget why they had made a given decision. Subjects would answer, "I don’t remember" if they realized they had forgotten, but just as often would apparently invent an incorrect answer that they thought they remembered.

Subjects frequently stated strategies that, if followed, would have selected different positions. Some of these are explained by satisificing, with subjects stopping when they found a "good enough" position and not noticing better alternatives. The randomness of this process was apparent during testing – the first tests asked subjects to recreate their training solutions and many subjects produced solutions that were similar in motivation but not detail.

When asked to describe the strategy used to generate a particular solution, some subjects got lost in reliving a problem and appeared to be explaining what they would do next time given the chance rather than what they actually did. This was particularly common when a subject felt the actual solution they proposed made them look bad. In Experiment 1 Phase 2 we did not tell subjects whether a perfect score was possible or discuss winning or losing a map in an effort to reduce this behavior.

Some responses were simply not true, although these might represent strategic lapses on the part of the subject. For example, some subjects would state that they always placed towers in symmetric pairs or avoided exterior walls but would occasionally break these rules. When these exceptions were pointed out, the subject would typically suggest that they had probably forgotten what their strategy was at that time. Given the amount of training and complexity of the task, subjects had likely not reached automaticity and were effortfully executing strategies, explaining the occasional lapses. While explainable, they still make the process of building and validating models difficult.

C.5.4 Unrecognized Vagueness

A common issue is that subject descriptions relied on concepts that were obvious to both the subject and the interviewer. It was only during implementation that we realized we lacked sufficient information to implement a strategy.

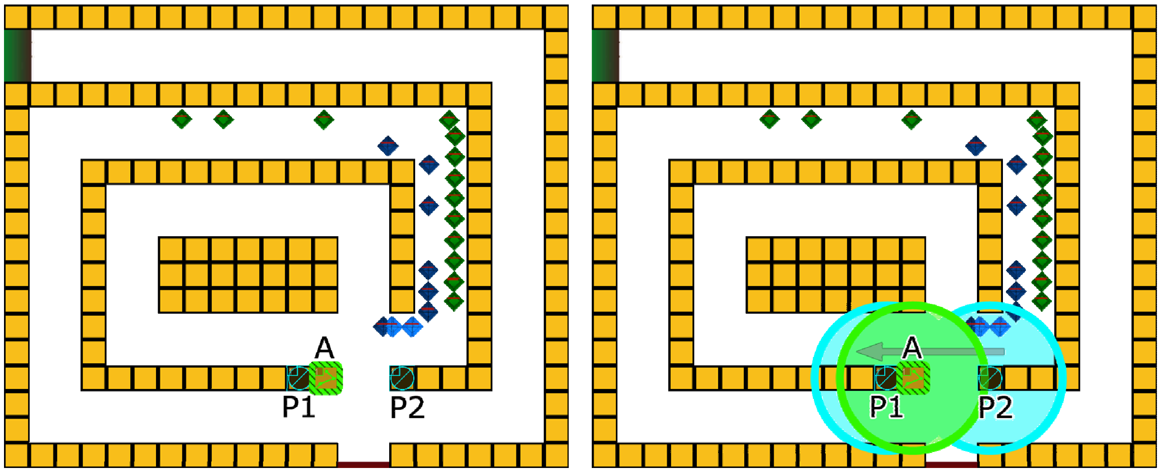


Figure C.4. Defining "Near". Placing a slowing tower "near" anchor tower A. LEFT: Position P1 is physically closer to A. RIGHT: Position P2 is more usefully near A. The goal of the slowing tower is to slow creeps as they enter A's range. Placing the tower at position P1 would slow the creeps as they leave A's range, too late to be effective.

Subject N0100 described their approach to problem solving as:

> "Place offense towers on areas that creeps pass by multiple times then place slowing towers ***near*** the offense towers so that the creeps are ***slow while in the offense towers' range***".

Figure C.4 shows an instantiation of this. In the image, an offense tower has been placed at position A, a position that creeps pass multiple times. They now want to place a slowing tower near the offense tower. There are two candidate positions, P1 and P2. Position P1 is adjacent to A while P2 is three tiles away. Objectively, P1 is nearer placement anchor A and thus a better position. However, this is not what the subject meant when they used the term "near". "Near" is shorthand for "usefully near". A slowing tower causes the creeps it attacks to move at half speed for two seconds. The slowing tower is placed near the offense tower so that the creeps move slowly through the offense tower's range, giving the offense tower more time to act. A slowing tower is near an offense tower if it causes the creeps to be slowed while in the offense tower's range. Position P2 accomplishes this. P1, in contrast, is closer but comes after the offense tower, causing the creeps to move slowly after they leave the offense tower's range, where slowing is no longer useful. P1 is nearer but only P2 is usefully near.

During the data capture, many of the terms used by subjects seemed unambiguous and were understood by both the subject and experimenter. Because the experimenter was familiar with the domain, it was not immediately recognized that terms such as "near" had a more complex and context-dependent definition that required knowing the direction of travel on the map, the properties of the slowing tower and the goals of the subject. There is no way to properly interpret this information without domain knowledge and an understanding of the subject's beliefs, goals and intentions.

Issues with unrecognized vagueness were not realized until the construction of the computer model. A computer model using an objective measurement of "near" only needed a simple distance measure. This was used in the first simulation agent, causing it to produce solutions that did not match the modeled subject's solutions either qualitatively or in performance. Correctly implementing "near" ended up being quite complex. We first needed to model the concept of a tower’s local neighborhood (as defined by the tower’s range) and then develop a means of determining whether two neighborhoods overlapped and in what way (i.e., did the slowing tower’s range overlap the front or back of an offense tower’s range as defined by path direction). A significant amount of work went into defining the basic spatial ideas necessary to model simple subject descriptions.

C.5.5 Terminology

There were two types of subjects, those that talked about the properties of a position (e.g., the position had the largest number of path cells in range) and those that talked about spatial structures (e.g., islands, corners, U-turns). The latter group described strategies such as "place towers at the end of a U-turn". In implementing these strategies, it became clear that these strategies simply didn't work. We expected some vagueness in subject descriptions and made small adjustments to compensate but in some circumstances the changes needed to make the reported strategies work altered the strategy to the point that they became fundamentally different strategies. The strategies reported by many subjects weren't simply vague or inaccurate; they led our investigation in the wrong direction.

Our mistake was in believing that, to our subjects, spatial features had objective definitions (i.e., a circle is a circle independent of who is asking or why they're interested). Although obvious in retrospect, when subjects reported their strategies in terms of spatial features (U-turns, corners, passes, islands, areas, corridors, etc.), what they actually meant is that their strategies depended on abstract, goal-relevant properties for which spatial terminology was a short cut both for explaining their reasoning and searching for relevant areas on the map. In goal-directed spatial reasoning, the subject is not interested in spatial features but spatial affordances -- the opportunities a spatial arrangement offers for a specific objective.

This issue proved to be an extremely important one and led to a number of insights that ultimately led to the creation of useful spatio-temporal reasoning agents. The topic is covered in more depth in Section 5.5.1 Features, Properties and Relationships vs. Affordances.

C.5.6 Benefits of Disadvantages

The problems we encountered in converting what humans said to what computers needed so as to be able to replicate their behavior helped us understand concepts we would not otherwise have discovered.

Wrestling with human data helped us understand problem-relevant domain intricacies that were not immediately apparent, even to domain experts. Attempting to replicate all strategies observed forced us to pay attention to the details differentiating each subject's set of strategies, revealing to us the enormous breadth of possible solutions and gave us insight into numerous variables necessary to accurately model spatial and strategic representations.

Most work on transfer learning in artificial intelligence focuses on the benefits of transfer learning but our use of human expert data combined with an investigation of subject learning histories revealed the significant (and currently unpredictable) dangers inherent in transfer learning, even in short distance transfer.

Human data helped us to realize that there were significant differences between spatial reasoning and goal-directed spatial reasoning and reinforced the idea of spatial affordances as being fundamentally different than spatial features. This was especially important as both spatial reasoning and features and goal-directed spatial reasoning and affordances use the same vocabulary to discuss very different things.

Finally, struggling with human data showed us the limits of relying on domain experts and helped us create a significantly more powerful behavioral model than we began with. It also made us realize that, even with a complex computer model of spatial reasoning in a small, constrained domain, we still cannot accurately represent many of the features necessary to generate (as opposed to merely describe) a specific individual's behavior in a manner detailed enough to be accurate and general enough to accurately simulate transfer learning. This in turn helps us appreciate the level of complexity in building data-driven models of human behavior and the size of the work left before us.

Appendix D

Detailed Statistical Analysis of Experiment 2

This chapter provides a more detailed look at the results from Experiment 2. As mentioned in Chapter 11 Results of Experiment 2, we are interested in which factors impact the problem solver's performance. We are specifically interested in the following factors:

- Domain Experience
    - Does experience with a related domain improve performance?
    - Does experience with a related domain decrease time-space conflation?
- Space-Time Conflation
    - Does treating space as a proxy for time outperform baseline strategies?
    - Does treating space and time as independent outperform treating space as a proxy for time?

- Human Ability vs. Computer
    - When using the same strategies, do humans, who presumably think more flexibly, outperform computers, which presumably are more accurate?

D.1 Definitions

This section defines some of the concepts needed to understand the following analysis.

- **Primary Strategies**. Subjects used many strategies but the majority used one of two sets of strategies, Advanced Maximum Usable Range (AMUR) or Differential Slowing (DS). Analyses based on strategies divide subjects into these two groups (subjects who used neither are excluded from those analyses).
- **Map Categories**. Maps were divided into three categories: normal, ceiling effect and DS amenable. These groupings were used to better analyze the data.
- **Significance**. We use the terms ***slight***, ***modest***, ***large*** and ***very large*** throughout the chapter. The precise meanings of these terms is explained here.
- **Difficulty**. Several maps exhibited a ceiling effect, making their analyses meaningless. To analyze these maps, the solutions were re-run on a harder difficulty level. The meaning of difficulty levels in GopherTD is discussed here.

D.1.1 Primary Strategies

Subjects were categorized as being predominantly Advanced Maximum Usable Range (AMUR) or Differential Slowing (DS) strategy users. The components of these strategies are explained in detail in Chapter 8 Strategies but we summarize the important points here.

AMUR is not technically a strategy but a pair of strategies, MAXIMUM USABLE RANGE and SLOW IN RANGE. MAXIMUM USABLE RANGE refers to any of the strategies in Section 8.1.1 To Increase Coverage Area, where the goal is to place the tower such that its range covers as much of the path area (i.e., the area where the creeps will be) as possible. It attempts to maximize the amount of space covered, making it a spatial strategy. STRATEGY TP1 SLOW IN RANGE says to place slowing towers around the offense towers so that the creeps spend more time in the offense towers' range. This strategy explicitly maximizes time, not space, meaning that AMUR is technically a hybrid spatio-temporal strategy, but one that focuses more heavily on space.

It is tempting to think that subjects using the same strategy (or, in this case, pair of strategies) would get the same score. We observed three reasons for why this didn't happen. The first is that subjects use more than these two strategies, and the additional strategies contribute to performance. Many of these are optimizing strategies that subjects create during testing. An example is STRATEGY E1 AVOID SINGLE-SIDED INTERIOR CORNERS. This strategy eliminates from consideration a specific class of suboptimal positions, reducing a type of waste that many subjects don't immediately notice. This also explains why some subjects felt that their performance increased with experience.

A second reason why subjects using the same strategy achieved different scores is that MAXIMUM USABLE RANGE is not a strategy but a family of strategies. While subjects attempted to maximize a tower's usable range, the definition of usable range varied between subjects. Subjects also varied in how they measured usable range. Details are provided in Section 8.1.1 To Increase Coverage Area. While all of these strategies attempt to maximize space, they differ in effectiveness. Note that seven of the spatial measurement strategies are satisficing strategies. In this domain, a computer agent can quickly determine which position(s) offer the MAXIMUM USABLE RANGE. As measured by a combination of answer quality and run-time performance, it is believed that these specific satisficing strategies used by humans underperform the optimal ones used by the computer under the current conditions. It is possible that the satisficing strategies would outperform the computer under other conditions, such as significantly larger maps, where a human's ability to visually pick out features such as U-turns might be significantly faster than measuring the value at every position as a non-satisficing computer would do.

The third reason why some subjects outperformed others using the same strategy is simple execution failure. Subjects sometimes didn't notice certain parts of the map and in at least one instance a subject conflated his strategies, accidentally applying a strategy meant for towers in one role to a tower in a different role. Even under laboratory testing conditions, subjects were sometimes distracted and as a result made mistakes.

As with MAXIMUM USABLE RANGE, DIFFERENTIAL SLOWING is a not a single strategy, it is a family of strategies. Details are provided in Section 8.2.2 To combat attack window decay Strategy Family: Differential Slowing, strategies TP9-TP15. We also include the related STRATEGY TP8 EXPLOIT GEOMETRY. Just as all of the MAXIMUM USABLE RANGE strategies involved maximizing space, all DIFFERENTIAL SLOWING strategies involve separating the two lines of creeps. A path synchronization gap between the two lines of creeps is created and/or exploited, with one line falling behind the other. In the best case, the two lines are fully separated, doubling the offensive towers' activation time; after their first attack opportunity, they get a second opportunity to attack when the second group comes through. It is often the case that the place where the two lines of creeps are most separated is not the place where the tower would achieve its MAXIMUM USABLE RANGE. In these cases, the problem solver obtains time by giving up space.

In summary, AMUR assumes that space and time are perfectly correlated and therefore the best way to increase a tower's activation time is to increase the tower's usable range. DS assumes that space and time are independent and that the increasing a tower's activation time sometimes means picking positions that decrease usable range. We therefore consider AMUR to be a predominantly spatial strategy and DS to be a predominantly temporal strategy.

Details for all strategies are given in Chapter 8 Strategies.

D.1.2 Map Categories

Experiment three used 16 maps. Subjects tended to group maps based on several non-mutually exclusive criteria, the predominant one being the map's amenability to a given strategy used by the subject. Because the strategy sets varied between subjects, there is no agreed upon way to categorize the maps. While it will not match how the subjects categorized them, in this section we classify the maps into groups intended to assist the reader in understanding the results reported in this chapter.

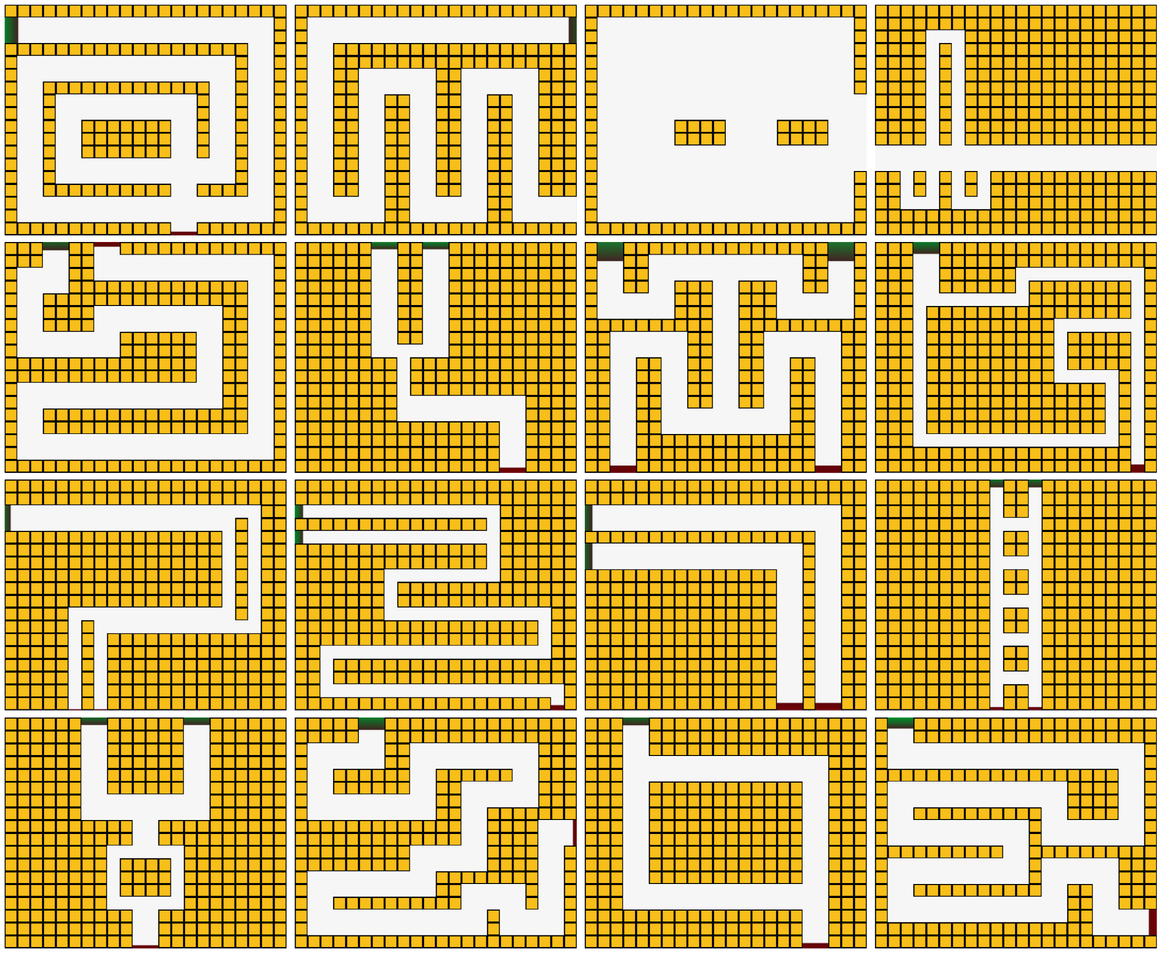


Figure D.1. Experiment 2 Maps. The 16 maps used in Experiment 2, in the order they were presented.

On five of the 16 maps tested (Figure D.2), almost all subjects received perfect or near-perfect scores. This suggests these maps were too easy, resulting in a ceiling effect that prevents us from being able to answer the performance questions posed in this chapter.

Many subjects noted that, on these maps, the walls were skinny and the path often doubled back, giving towers multiple opportunities to attack. Note that the latter almost necessitates the former – because each map is a fixed size (18x22), unless the path is unusually short, in order for the path to double back on itself the width of the walls must be decreased. For most of these maps, there were very few positions where one could put a tower and not be able to attack. As a result, even the random baseline AI did well on these maps. Many subjects reported that they spent relatively little effort on these maps because they knew they didn't need to. Appendix F Solutions discusses the topic of map properties and difficulty in more depth.

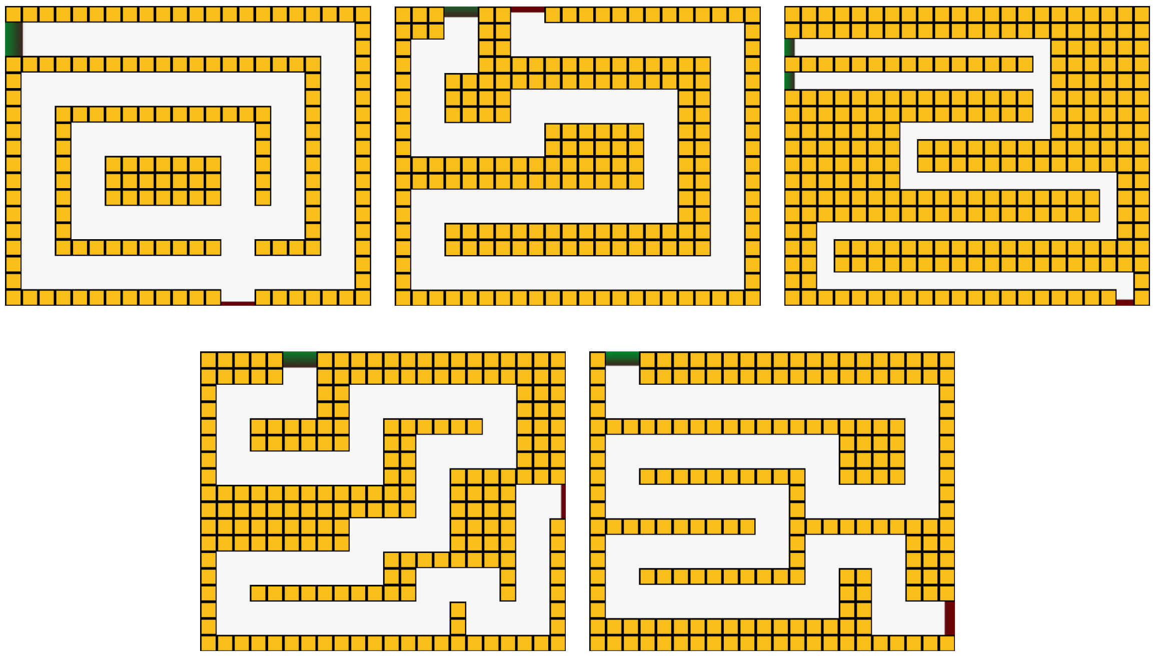


Figure D.2. Ceiling Effect Maps. Almost all subjects achieved maximum scores on these maps. Maps: RoundTheTwist, Elemental-ish, Slim Pickings, Snaking Path, Switchback.

As explained in Section D.1.1 Primary Strategies, DS strategies exploit path synchronization gaps. Normally two lines of creeps are synchronized, moving side by side, but when moving around corners, the creeps on the outer path have a further distance to travel and thus fall behind, creating a path synchronization gap. In some instances, the two lines of creeps will follow paths that are not only not side-by-side but are radically different in length (Finger and Void; see Figure D.3, top row). Gaps can also be created, grown or shrunk using slowing towers. This creates an affordance of value to DS users but not AMUR users. We grouped five maps (Figure D.3) based on properties related to path gaps.

On two maps (Finger, Void), one path was significantly longer than the other, creating a significant path gap. Because subjects rarely noticed a map property unless it was used by their strategies, subjects that did not use path gap strategies did not normally notice path gaps. However, for these two maps, the gaps were so large that almost all subjects noticed them. Throughout our research we have noticed that subjects that recognized an affordance were almost always able to quickly form a strategy to exploit them. Correspondingly, many (although not all) subjects that did not normally use the EXPLOIT GEOMETRY or DS strategies did so on this map.

Two of the maps (Pathways, No Left Turns) had significant but less obvious path gaps. The final map (Do the Splits) had no path gap. Only subjects that normally used the DS strategy used it on these maps. Subjects that did not use the DS strategy reported these maps as being exceedingly difficult, if not impossible, to solve. Subjects that used the DS strategy reported that these maps were no harder than any other. These maps are an excellent example of how the difficulty of a problem is not an absolute or inherent property of the problem but a subjective one defined by its relationship to the problem solver.

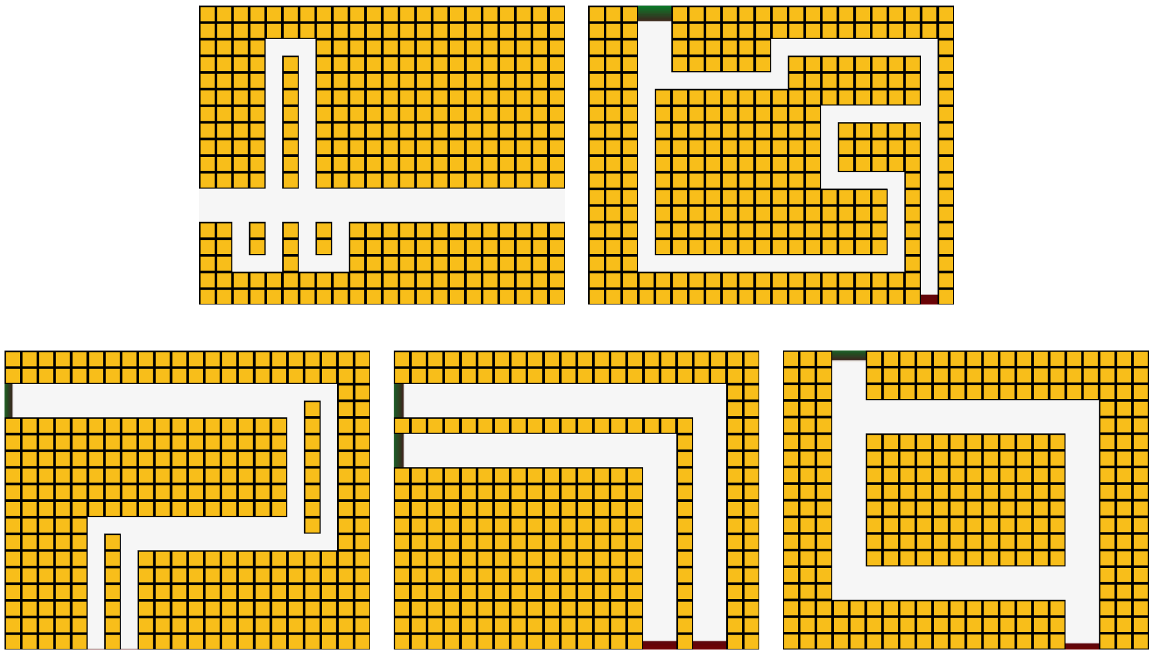


Figure D.3. Maps Amenable to Differential Slowing Maps. These maps either suggest, are amenable to or can only be solved by a DIFFERENTIAL SLOWING strategy. Top Row: Finger, Void. Bottom Row: Pathways, No Left Turns, Do the Splits.

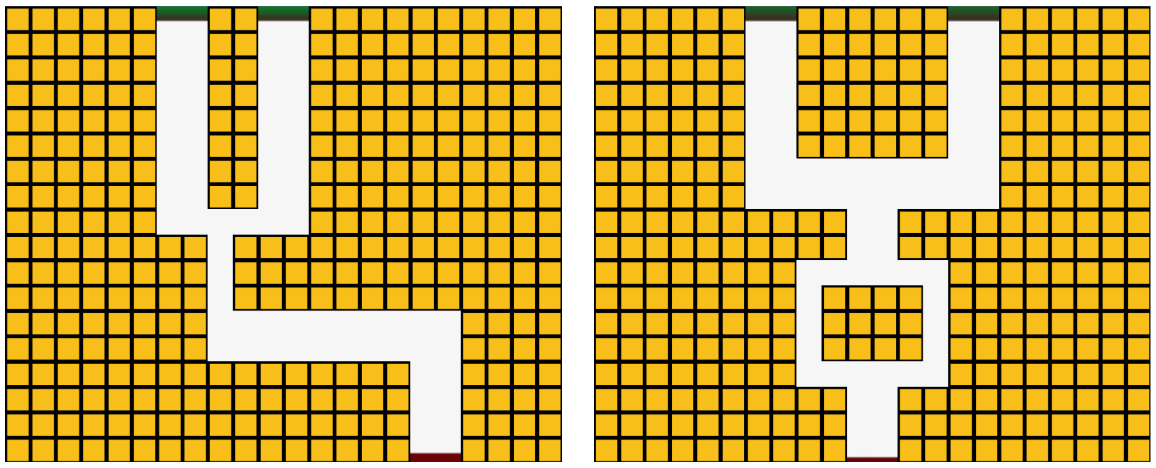


Figure D.4. Hard Maps. These maps were hard for all subjects. Maps: 2 In 1 Out, Bottleneck.

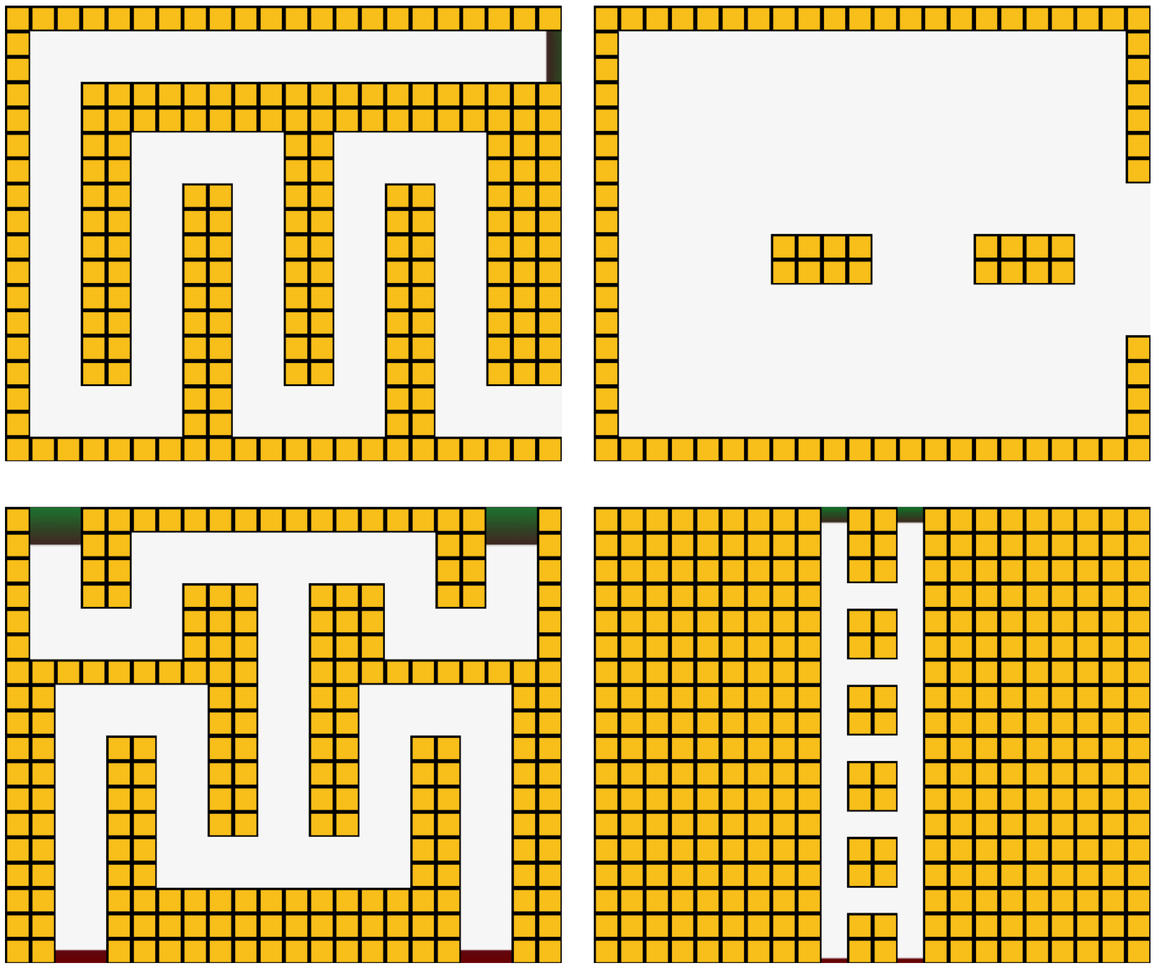


Figure D.5. Other Maps. Maps that do not fall into the other categories. Maps, left to right: Up 'n Down, No Support, The Frog, Ladder.

All subjects, regardless of which strategies they used, stated that two of the maps (2 In 1 Out, Bottleneck) were very difficult. Many subjects classified these maps as "fast maps" or "short maps" as the paths were shorter than on many other maps. There were no loop backs as in the easier maps and the limited amount of time to slow one line before the creeps merged made the DS strategy relatively ineffective. Again, Appendix C - Perfect Scores discusses the topic of map properties and difficulty in more depth.

The remaining four maps (Up 'n Down, No Support, The Frog, Ladder) were tractable for all subjects without being so easy that a ceiling effect removed their discriminative power.

It is worth noting that on one map (Ladder), almost all subjects generated similar solutions. This appears to be because the problem is not particularly amenable to any strategy. We initially concluded that this map had no discriminative power until one of the domain novices used a strategy that was at first glance exceptionally counter-intuitive but which proved to be highly effective. The solution to this map is presented in Appendix C - Perfect Scores.

D.1.3 Significance

Using the assigned towers in Experiment 2 and the 11 maps with discriminatory power (i.e., excluding the five easiest maps as they demonstrate ceiling effects), random placement of towers earns an average score of 9.19. Domain novices earn an average score 20.56. The maximum score on a map is 28. With a 19 point range from the average random score to the maximum score and a 7.5 range between the average novice score and the maximum score, a two point difference between scores is fairly important. Table D.1 shows the labels we use for different values.

Range	**Magnitude**
< 1.00	Essentially no difference
1.00 - 1.99	Slight
2.00 - 3.99	Modest
4.00 – 7.99	Large
>= 8.00	Very Large

Table D.1. Qualitative Rating for Differences in Scores.

Where we use percentages, numbers refer to differences in percentage points, i.e., the difference divided by the maximum score. This allows us to state the magnitude without concern for whether we are reporting a gain or loss. For example, given the scores A=15 B=25, the 10 point difference can be reported as B scoring 67% higher than A (10/15) or A scoring 40% lower than B (10/25). These scores are also sensitive to the size of the base. For example, given the scores A=8 B=18, there is still a 10 point difference but now that difference is reported as B scoring 125% higher than A (10/8) or A scoring 56% lower than B (10/18).

To avoid this sensitivity to direction and base, we report differences as normalized percentage points, with a one point difference on a 28 point scale always being worth 3.57%. In the above examples, B outperformed A by 36% and A underperformed B by 36% regardless of whether the scores are 15 and 25 or 8 and 18. For this experiment, this method of reporting often has the effect of lowering the perceived gain or loss. We feel that the ability to make accurate comparisons outweighs the emotional benefit of seeing larger impact numbers.

D.1.4 Difficulty Settings

GopherTD is based on the commercial game Vector TD. In Vector TD, the player must survive 50 waves of creeps. The only difference between waves is the health of the creeps. At level 1, creeps have 50,000 points of health. This increases by a compounded 10% per level to a maximum of 2,262,963 at level 50. As the health of the creeps increase, the number of times they must be shot by a tower increases and thus the required active time of the set of towers increases.

Experiments 1 and 2 were equivalent to Vector TD's level 22, where creeps had 370,012 health points. To eliminate ceiling effects on easier maps, we also ran solutions post hoc at level 27, where creep health increased by 61% to 595,909.

D.2 Effects of Domain Experience

One of the questions we wished to answer was whether experience on conceptually similar tasks would profitably transfer to a novel domain. None of the subjects had previous experience with Vector TD, the game our task is based on, but 10 of the 20 subjects in Experiment 2 had experience with one or more other tower defense games. We wished to know whether that experience would help them in this task.

Tower defense games have similar goals and relatively similar mechanics but are quite different in the details. In terms of magnitude, the difference between two games in the same sub-genre of tower defense is often similar to the difference between racquet ball and tennis – they are not the same game and there are strategies that work in one but not the other but they have more in common than not. There is no objective, agreed upon way to measure transfer distance but Haskell's (Haskell 2000) five categories of transfer model is a common metric. Under this scale, we believe transfer across two different tower defense games would qualify as both ***strategic*** and ***theoretical knowledge transfer***. Haskell's model, like many others, only recognizes two distance levels, ***near*** and ***far***. We have argued elsewhere (unpublished) that abstraction, generalization and transfer distance exist on a sliding scale. We believe that transfer across tower defense games falls somewhere in the middle.

Subjects were placed into the categories Domain Novice (n=10) and Domain Experienced (n=10). The distinction was binary – for subjects with domain experience, we did not account for amount of experience, either in number of years or hours per week, nor did we account for their skill level (not everyone with experience is an expert). We also did not separate subjects by the number of different tower defense games or types of tower defense games they had experience with. The sole criterion was whether the subject had any experience with tower defense.

It should be noted there are many different types and sub-types of tower defense games. Many are similar enough that the transfer distance is not significant but in some cases, especially across types of tower defense games, the distance can be relatively large. The most notable example of this is the "mazing" genre, in which subjects design their own maps. Players are given a blank map and place their towers to create mazes. Some of the skills and strategies in this variant are relevant to the task in this experiment, many are not. Many subjects had experience with this type of tower defense but not the one used in this experiment. It should also be noted that transfer distance does not necessarily correlate with performance. In at least one case (in Experiment 1, not the experiment discussed here), a subject performed at chance levels because of negative transfer from a game that was similar to the one being tested.

It must also be noted that none of the subjects in the domain novice category had any significant experience with any other type of game other than casual games such as solitaire. We did not encounter any subject who played games but had not played tower defense. For this reason, we cannot determine whether the effects of domain experience demonstrated here are due to experience with games in general or tower defense specifically.

It is well known that there are difficulties in quantitatively studying complex task performance due to issues such as the number of variables, and it is difficult to use human subjects in trainable tasks as there is no way to accurately know their learning performance. Despite these complications and the lack of granularity used dividing subjects into groups, we believe that the data presented here shows a strong effect of experience.

Although it is a tangential issue, it is our belief that the existing transfer scales do not accurately capture how transfer occurs. Rather than transferring a block of knowledge a given distance, near or far, our research suggests that, at least in this domain, transfer is a matter of percentage – how much of what you know in one domain can be used in a new domain and how much knowledge is needed in the new domain. The dissimilarity of the mazing and non-mazing genre of tower defense (GopherTD is non-mazing) suggests moderately high transfer distance but viewed another way, transfer of tower selection, upgrade and combination strategies from a mazing to a non-mazing is near transfer; transfer of strategies for designing mazes is far transfer and nothing from the mazing domain can be transferred to the portion of our task that involves analyzing a given map. From the perspective of the reasoning by strategy approach to problem solving that we advocate, the relevant issue here isn't how far knowledge can be transferred and then applied but how much can be transferred and how that is integrated with new knowledge.

D.2.1 Impact of Experience on Performance

Our initial analysis compared domain novices to domain experienced across all maps. A 2 (experience level) x 16 (map) mixed factor ANOVA analysis showed:

- a main effect of map, F(15, 270) = 67.36, MSE = 6.42, $p < .001$
- a map x group interaction, F(15, 270) = 3.42, MSE = 6.42, $p < .001$
- a main effect of group, F(1, 18) = 19.92, MSE = 9.57, $p < .001$

Across all maps, experienced players (M = 24.21) outperformed novice (M = 22.66) players. With a maximum possible score was 28, the 1.55 point gap means the domain experienced subjects outperformed domain novices by 5.5%. The difference is statistically significant, meaning there is a consistent difference between the groups, but the magnitude of that difference is relatively minor. However, as explored below, experience differences may have been obscured by ceiling effects for several of the maps.

The maps had a substantial impact on a subject's score with a large range in the performance scores by map. This is to be expected – the maps were designed to have different difficulty levels. The average score for each map, independent of group, is shown in Table D.2.

Map	**Mean**	**SD**	**Min**	**Max**
DoTheSplits	13.90	3.48	8	21
NoLeftTurns	15.55	4.50	11	28
2in1out	17.95	2.54	14	23
Pathways	19.10	5.92	10	28
Bottleneck	20.35	2.06	17	26
Void	22.80	3.25	18	28
Ladder	23.70	2.08	20	27
TheFrog	25.25	2.63	20	28
NoSupport	25.65	2.54	17	28
Finger	26.15	1.90	22	28
UpAndDown	26.75	2.88	15	28
SlimPickings	27.10	2.49	17	28
Elemental_ish	27.20	1.64	21	28
SnakingPath	27.50	0.69	26	28
RoundTheTwist	28.00	0	28	28
Switch	28.00	0	28	28

Table D.2. Average Score per Map. The maximum score for each map is 28. Highlighted maps have averages near the maximum score.

Five of the 16 maps exhibit a ceiling effect, with average scores at or near the maximum (means across groups of 27 or higher). If a map is so easy that everyone does well on it, there is no way to tell if domain experienced subjects have better solutions than domain novices. This obscures the actual difference between domain novices and experienced.

To get a more accurate picture, we re-ran the analysis with the ceiling effect maps removed. A 2 (group: experienced or novice) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(10, 180) = 48.27, MSE = 8.28, p < .001
- a map x group interaction, F(10, 180) = 3.45, MSE = 8.28, p < .001
- a main effect of group, F(1, 18) = 15.96, MSE = 13.91, p = .001

After excluding the maps with ceiling-level performance, experienced players (M = 22.56) still outperformed novice (20.56) players. The magnitude of the difference rose slightly from 5.5% to 7%. As before, the main determinant of score was the map. Performance ranged from a mean of 13.9/28 (approximately 50%) to 26.75/28 (96%). Scores are shown in Table D.3.

While scores varied by map, they did not vary in the same way for each group. The reason for the significant map x group interaction is that experienced players clearly outperformed novice players on some maps but not on others. As shown in Table D.3, one-way ANOVAs performed on each map separately showed that experienced players demonstrated a clear and significant advantage on two maps (Pathways, Void), a modest but reliable advantage on two other maps (DoTheSplits, 2in1out) and a nonsignificant advantage for the remaining maps, except for two maps that showed instead a slight novice player advantage or no difference (NoSupport, Finger).

GopherTD, our experimental tool, is based on Vector TD, a popular commercial tower defense game. In that game, there are 50 difficulty levels, each level differentiated by how difficult it is to destroy a creep (creeps have more health on higher levels). Experiment three was equivalent to difficulty level 22. At this difficulty level, ceiling effects were obtained on five maps. We investigated what might have happened had we run the test at a higher difficulty level.

We conducted a simulation of how the subject solutions would have performed at level 27, a level at which we saw no ceiling effects. Subjects were not directly involved in this test. We used the solutions generated by subjects in the original test.

Map	Mean (SD) Exper.	Mean (SD) Novice	Diff.	Comparison	Significance
NoSupport	24.9 (3.41)	26.4 (0.84)	-1.5	Slight novice advantage	Nonsig.
Finger	26.0 (2.00)	26.3 (1.89)	-0.3	Essentially no difference	Nonsig.
Ladder	24.0 (1.63)	23.4 (2.50)	0.6	Essentially no difference	Nonsig.
TheFrog	25.6 (2.68)	24.9 (2.69)	0.7	Essentially no difference	Nonsig.
UpAndDown	27.4 (0.84)	26.1 (3.99)	1.3	Slight experienced advantage	Nonsig.
Bottleneck	21.1 (1.97)	19.6 (1.96)	1.5	Slight experienced advantage	p < .11
2in1out	19.1 (2.69)	16.8 (1.87)	2.3	Modest experienced advantage	p = .039*
NoLeftTurns	17.1 (5.78)	14.0 (2.00)	3.1	Modest experienced advantage	p < .13
DoTheSplits	15.5 (3.89)	12.3 (2.16)	3.2	Modest experienced advantage	p = .036*
Void	24.7 (3.23)	20.9 (1.97)	3.8	Modest experienced advantage	p = .005**
Pathways	22.8 (5.37)	15.4 (3.84)	7.4	Large experienced advantage	p = .002**

Table D.3. Experience Advantage per Map. Sorted by the mean difference between experienced and novice players. SD = Standard Deviation; Exper. = experienced; * indicates p < .05, ** indicates p < .01.

Re-run at a higher difficulty level, 2 (experience level) x 16 (map) mixed factor ANOVA analysis showed:

- a main effect of map, F(5.23, 94.15) = 98.11, MSE = 19.92, p < .001
- a marginal map x group interaction, F(5.23, 94.15) = 2.06, MSE = 19.92, p =.074
- a main effect of group, F(1, 18) = 4.75, MSE = 24.55, p =.043

Across all maps, experienced players (M = 15.23) significantly outperformed novice (M = 14.02) players. The 1.21 point gap represents a 4% advantage, a relatively small but statistically significant difference. As before, the main determinant of score was the map. Performance ranged from a mean of 5.65/28 (approximately 20%) to 25.28/28 (90%).

The main effect of map at the higher level of difficulty again reflected a wide range in overall levels of performance across the maps. Table D.4 presents the maps ordered by performance level across all participants from lowest to highest. The relative difficulty of each map has changed slightly but the maps that originally displayed ceiling effects still make up five of the six easiest maps.

Map	Mean	SD	Min	Max
DoTheSplits	5.65	1.31	3.75	8.75
NoLeftTurns	6.84	2.90	3.75	15.25
2in1out	7.96	1.32	6.00	11.75
Bottleneck	9.00	1.82	6.75	13.25
Pathways	9.11	3.36	4.50	18.00
Ladder	11.19	1.14	9.75	13.50
TheFrog	13.10	2.42	9.50	17.25
Void	15.01	3.54	7.75	21.50
NoSupport	17.31	3.00	6.25	20.00
UpAndDown	17.68	4.25	5.00	25.50
Elemental_ish	17.85	4.79	9.00	28.00
SlimPickings	18.01	3.61	9.00	25.50
SnakingPath	18.34	3.64	14.50	28.00
Finger	19.13	1.53	17.00	23.00
Switchback	22.56	2.71	17.25	28.00
RoundTheTwist	25.29	2.70	19.25	28.00

Table D.4. Average Score per Map (Advanced Difficulty). Sorted by mean. Each participant was simulated four times, and then the average score across the four simulations for each participant across all 20 participants for each map was calculated. SD = Standard Deviation; Min = Minimum mean score per participant; Max = Maximum mean score per participant.

We wanted to answer the question, does experience with a related domain improve performance? The data indicates that, averaged across the maps used in this experiment, the answer is yes, but not to a large degree. At the simulated higher level of difficulty, experience-related differences are slight or modest, with statistically significant differences observed for two maps that were hard for all participants (Pathways and Bottleneck) and also for a map that earlier showed ceiling effects (Elemental-ish).

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Exper.	**Novice**			
NoSupport	16.50 (4.01)	18.13 (1.19)	-1.63	Slight novice advantage	Nonsig.
RoundTheTwist	24.80 (3.14)	25.78 (2.22)	-0.98	Essentially no difference	Nonsig.
Finger	18.73 (1.80)	19.53 (1.18)	-0.80	Essentially no difference	Nonsig.
DoTheSplits	5.68 (1.57)	5.63 (1.07)	0.05	Essentially no difference	Nonsig.
Switchback	22.65 (3.03)	22.48 (2.51)	0.18	Essentially no difference	Nonsig.
Ladder	11.53 (1.27)	10.85 (0.93)	0.68	Essentially no difference	p = .19
2in1out	8.43 (1.68)	7.50 (0.65)	0.93	Essentially no difference	p = .12
SlimPickings	18.53 (3.29)	17.50 (4.01)	1.03	Slight experienced advantage	Nonsig.
SnakingPath	18.95 (4.81)	17.73 (2.00)	1.23	Slight experienced advantage	Nonsig.
Bottleneck	9.80 (2.20)	8.20 (0.86)	1.60	Slight experienced advantage	p = .046*
TheFrog	13.93 (2.81)	12.28 (1.73)	1.65	Slight experienced advantage	p = .13
NoLeftTurns	7.78 (3.74)	5.90 (1.36)	1.88	Slight experienced advantage	p = .15
Void	16.05 (3.75)	13.96 (3.15)	2.09	Modest experienced advantage	p = .195
UpAndDown	19.13 (3.55)	16.23 (4.56)	2.90	Modest experienced advantage	p = .13
Pathways	11.08 (3.57)	7.15 (1.61)	3.93	Modest experienced advantage	p = .005**
Elemental_ish	20.15 (3.99)	15.55 (4.57)	4.60	Large experienced advantage	p = .027*

Table D.5. Average Score per Map, By Experience (Advanced Difficulty). Sorted by magnitude of experience advantage. SD = standard deviation; Exper. = experienced; * indicates p < .05, ** indicates p < .01.

Condition	Mean Experienced	Novice	Experienced Advantage
Non-Ceiling Maps	22.56	20.56	7.1%
All Maps	15.23	14.02	4.3%
Advanced Difficulty			

Table D.6. Comparison of Subjects by Experience.

D.2.2 Impact of Experience on Strategy Selection

Of the 10 subjects with no previous domain experience, 70% used AMUR and 30% used other strategies (Strategy SP4: Maximum Percent of Path, Strategy SP5: Maximum Number of Passes and Strategy TP16: KILL ZONE JUMPING). No domain novice used DS.

Of the 10 subjects with tower defense experience, 40% used AMUR, 50% used DS and 10% used another, more advanced, temporal strategy (Strategy TP5: TEMPORAL ATTACK WINDOW SEPARATION).

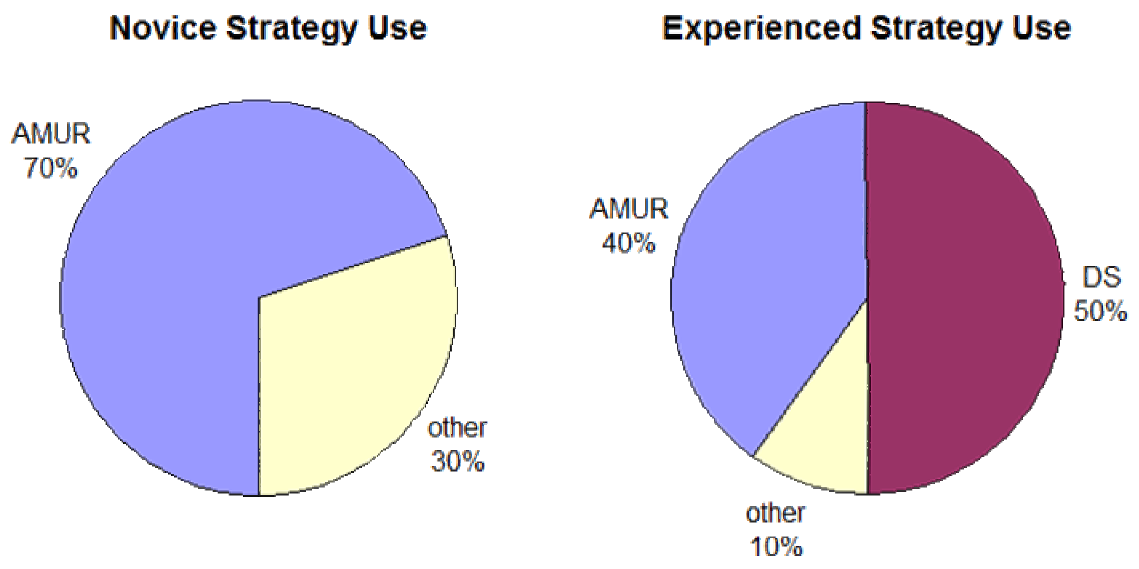


Figure D.6. Subject Strategy Selection as a Function of Domain Experience.

While having domain experience did not guarantee that the subject used a strategy that explicitly maximized time, not having domain experience did seem to ensure that they did not. Domain novices maximized space, strongly suggesting that they converted temporal problems into spatial ones.

D.2.3 Impact of Experience on Strategy Execution

As revealed in the previous section, no one in the domain novice group used the DS strategy but subjects in both the domain novice and experienced groups used the AMUR strategy. Do subjects with experience with other tower defense games but not the one being tested use the AMUR strategy better than those with no tower defense experience?

We began by looking at the 11 non-ceiling maps. A 2 (group: experienced AMUR or novice AMUR) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, $F(10, 90) = 30.51$, $MSE = 7.94$, $p < .001$
- no map x group interaction, $F < 1$
- a main effect of group, $F(1, 9) = 4.97$, $MSE = 5.38$, $p = .053$

Across all maps, subjects with domain experience that used the AMUR strategy (M = 21.43) outperformed domain novices using the same strategy (M = 20.46). The 0.97 point gap represents a 3.4% advantage for experienced players. Experienced AMUR players numerically outperformed novice AMUR players on all but three non-ceiling-level maps (Bottleneck, NoLeftTurns, NoSupport), with the largest numerical experience-related differences (within the AMUR strategy) found for maps DoTheSplits, Void, and Pathways; however, follow-up one-way ANOVAs on each of the maps separately showed that none of these individual map differences were significant.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Exper. AMUR	**Novice AMUR**			
NoSupport	24.50 (5.07)	26.43 (0.98)	-1.93	Slight Novice AMUR advantage	Nonsig.
Bottleneck	19.75 (0.50)	20.00 (2.16)	-0.25	Essentially no difference	Nonsig.
NoLeftTurns	13.50 (2.08)	13.57 (2.07)	-0.07	Essentially no difference	Nonsig.
2in1out	17.75 (2.50)	17.14 (2.12)	0.61	Essentially no difference	Nonsig.
Finger	26.75 (1.50)	26.14 (2.19)	0.61	Essentially no difference	Nonsig.
TheFrog	24.75 (3.78)	24.14 (2.80)	0.61	Essentially no difference	Nonsig.
Ladder	24.25 (1.71)	23.14 (2.19)	1.11	Slight Exper. AMUR advantage	Nonsig.
UpAndDown	27.25 (0.96)	25.43 (4.69)	1.82	Slight Exper. AMUR advantage	Nonsig.
DoTheSplits	14.50 (2.52)	12.43 (2.44)	2.07	Modest Exper. AMUR advantage	p = .213
Void	23.75 (4.03)	20.71 (2.06)	3.04	Modest Exper. AMUR advantage	p = .126
Pathways	19.00 (5.83)	15.86 (2.55)	3.14	Modest Exper. AMUR advantage	Nonsig.

Table D.7. Average Score per Map for Humans Using the Advanced Maximum Usable Range Strategy (Normal Difficulty). Sorted by mean experience advantage.

As before, we repeated the analysis using all 16 maps at the higher difficulty level. A 2 (experience level) x 16 (map) mixed factor ANOVA analysis on the Level 27 data for only those humans who adopted an AMUR strategy showed:

- a main effect of map, F(4.29, 38.64) = 54.76, MSE = 22.80, p < .001
- no map x group interaction, F(4.29, 38.64) = 1.70, MSE = 22.80, p = .17
- no main effect of group, F(1, 9) < 1

For those humans who adopted an AMUR strategy, at the Level 27 difficulty, there was little difference between the experienced players (M = 14.22) and the novice players (M = 13.74).

Map	Mean	SD	Min	Max
DoTheSplits	5.70	0.89	4.25	7.00
NoLeftTurns	5.89	1.33	3.75	7.75
2in1out	7.48	0.87	6.00	9.00
Pathways	7.84	1.80	5.50	11.25
Bottleneck	8.09	1.06	6.75	10.25
Ladder	10.81	1.10	9.75	13.25
TheFrog	12.14	2.67	9.50	17.25
Void	13.51	3.19	7.75	17.25
Elemental_ish	15.89	4.13	9.00	22.00
UpAndDown	16.98	5.21	5.00	25.50
SnakingPath	17.02	1.63	14.75	20.00
NoSupport	17.14	3.80	6.25	19.75
SlimPickings	17.32	3.47	9.00	22.00
Finger	19.41	1.15	17.50	21.75
Switchback	22.45	2.65	17.25	26.25
RoundTheTwist	24.89	2.77	19.25	28.00

Table D.8. Average Score per Map for Humans using the Advanced Maximum Usable Range Strategy (Advanced Difficulty). Sorted by mean. Each participant's solution was run four times to average out numerical instability. SD = Standard Deviation; Min = Minimum mean score per participant; Max = Maximum mean score per participant.

For humans who adopted an AMUR strategy, at the simulated higher level of difficulty, the maps were ordered as shown in Table Table D.8.

We were unable to test the impact of experience on transference of the temporal DS strategy as no novice subject used it. Do subjects with experience with other tower defense games but not the one being tested use the spatial AMUR strategy better than those with no tower defense experience? Just looking at the maps lacking a ceiling effect, the answer is yes, but the difference is not large (0.97 points) and of relatively low significance (p = .053). When all maps are considered and simulated at a higher difficulty level, the answer is no.

	Mean		Experienced
Condition	**Exper. AMUR**	**Novice AMUR**	**Advantage**
Non-Ceiling Maps	21.43	20.46	3.5%
All Maps Advanced Difficulty	14.22	13.74	not significant

Table D.9. Comparison of Subjects by Experience (AMUR).

In this instance, it does not appear that experience with a different set of tower defense puzzles helps ***if*** the subject uses the spatial strategy AMUR. As noted in the previous section, experience with a similar domain has a significant impact on the strategies that are used and, as will be shown shortly, that has a significant impact on performance.

D.3 Effects of Spatio-Temporal Independence

We have mentioned several times in this work that the goal in tower defense is temporal – maximize the amount of time the towers are active (this is a slight simplification; see Sections 8.4.2 To Convert Wounds to Kills and 8.4.3 To Reduce Overage for details). While many of the subjects in the experienced group used strategies that explicitly maximized the amount of time a tower was active, 40% of subjects in the experienced group and 70% of subjects in the novice group chose to maximize the amount of space the tower covered. They converted a temporal problem into a spatial one, implicitly assuming that the time and space were equivalent, or at least strongly correlated. In this view, increasing the amount of space a tower covers is the same thing as increasing the amount of time it will operate.

We learned from Experiment 1 that subjects conflated space and time in this domain. One of the goals of Experiment 2 was to determine whether this was a good thing.

Subjects were divided into two groups, those who used the ADVANCED MAXIMUM USABLE RANGE (AMUR) spatial strategy and those who used the DIFFERENTIAL

SLOWING (DS) temporal strategy (for more information, see Section 10.2 Measured Strategies). We asked two questions:

- Does using space as a proxy for time work?
- If so, does treating space and time as independent work better?

To answer the first question, we compared the performance of the members of the AMUR (spatial) group to three baseline strategies we created. The agents and the strategies they implemented were, in order of increasing sophistication:

- **AI_Random**. Place towers at any legal position.
- **AI_PathAdjacent**. Place towers on any legal position that is immediately next to a path tile.
- **AI_MUR**. The MAXIMUM USABLE RANGE strategy. Place each tower where it covers the most space. This strategy differs from the AMUR strategy only in that this places slowing towers where they cover the most range rather than next to the offense towers.

Each agent attempted each map 100 times. Agents used the exact same towers (four fast offense towers, 20 slowing towers) as the human subjects.

Before discussing the results we wish to mention that ANOVA measures rely on an assumption of sphericity, a measure of independence for variables in a within-subject study. When this assumption is violated, it potentially inflates the number of Type 1 (false positive) errors and therefore causes the F-ratio to appear higher than it really is, potentially making insignificant results appear significant. For many of the following tests, there was a statistically significant result for Mauchly's test of sphericity, indicating the sphericity assumption was violated. We compensated for these situations by applying the Greenhouse-Geisser correction, a common and conservative correction; where necessary, the degrees of freedom reported in the following analyses reflect this correction.

D.3.1 Effectiveness of Spatial-Proxying

Tower defense has a temporal goal but many subjects treat the problem as a spatial one and use spatial strategies. Given that the wrong property is being maximized, how well does spatial-proxying work?

We first compared the AMUR group to each baseline agent on the 11 maps that did not display a ceiling effect. We began with the baseline agent AI_Random. A 2 (Human AMUR vs. AI_Random) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(8.37, 912.12) = 57.80, MSE = 20.78, p < .001
- a group x map interaction, F(8.37, 912.12) = 28.32, MSE = 20.78, p < .001
- a main effect of group, F(1, 109) = 795.56, MSE = 18.50, p < .001

Across all maps, AMUR users (M = 20.81) significantly outperformed AI_Random (M = 9.19) players. The 11.62 point gap represents a 41.5% advantage. Human AMUR very clearly outperformed AI_Random on 10 of the 11 non-ceiling maps. Human AMUR and AI_Random performed nearly identically on one map, UpAndDown. Based on scores from the initial experiment, UpAndDown is the easiest of the non-ceiling effect maps.
The table below gives the mean performance for each of the 11 maps, ordered by the magnitude of Human AMUR advantage over the baseline agent AI_Random, with the statistical outcomes from one-way ANOVAs performed on each map separately.

We next compared the AMUR group to AI_PathAdjacent. A 2 (Human AMUR vs. AI_PathAdjacent) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.96, 868.10) = 73.36, MSE = 12.69, p < .001
- a group x map interaction, F(7.96, 868.10) = 6.59, MSE = 12.69, p < .001
- a main effect of group, F(1, 109) = 20.65, MSE = 11.17, p < .001

Across all maps, Human AMUR users (M = 20.81) outperformed AI_PathAdjacent (M = 19.36). The 1.45 point gap represents a 5.2% advantage. Human AMUR numerically outperformed AI_PathAdjacent on some but not all maps and significantly outperformed it for three maps (Finger, TheFrog, Void).

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**AI Random**			
UpAndDown	26.09 (3.78)	25.78 (2.69)	0.31	Essentially no difference	Nonsig.
DoTheSplits	13.18 (2.56)	8.18 (3.53)	5.00	Large human AMUR advantage	p < .001***
Pathways	17.00 (4.07)	9.82 (4.60)	7.18	Very large human AMUR advantage	p < .001***
Void	21.82 (3.13)	14.25 (4.84)	7.57	Very large human AMUR advantage	p < .001***
TheFrog	24.36 (3.01)	16.60 (3.60)	7.76	Very large human AMUR advantage	p < .001***
NoLeftTurns	13.55 (1.97)	4.74 (4.31)	8.81	Very large human AMUR advantage	p < .001***
2in1out	17.36 (2.16)	3.44 (4.35)	13.92	Very large human AMUR advantage	p < .001***
Bottleneck	19.91 (1.70)	3.86 (4.22)	16.05	Very large human AMUR advantage	p < .001***
Finger	26.36 (1.91)	8.19 (6.09)	18.17	Very large human AMUR advantage	p < .001***
Ladder	23.55 (2.02)	3.06 (3.97)	20.49	Very large human AMUR advantage	p < .001***
NoSupport	25.73 (3.04)	3.16 (4.23)	22.57	Very large human AMUR advantage	p < .001***

Table D.10. Average Score per Map for Humans Using the ADVANCED MAXIMUM USABLE RANGE **Strategy Compared to a Random Benchmark Computer Agent**. Sorted by magnitude of human AMUR advantage. *** indicates p < .001.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**AI PathAdj.**			
Pathways	17.00 (4.07)	18.28 (3.37)	-1.28	Slight AI PathAdj. advantage	Nonsig.
DoTheSplits	13.18 (2.56)	13.65 (2.24)	-0.47	Essentially no difference	Nonsig.
NoLeftTurns	13.55 (1.97)	13.91 (2.46)	-0.36	Essentially no difference	Nonsig.
UpAndDown	26.09 (3.78)	26.17 (2.47)	-0.08	Essentially no difference	Nonsig.
2in1out	17.36 (2.16)	17.21 (2.91)	0.15	Essentially no difference	Nonsig.
NoSupport	25.73 (3.04)	25.36 (2.91)	0.37	Essentially no difference	Nonsig.
Ladder	23.55 (2.02)	22.89 (2.81)	0.66	Essentially no difference	Nonsig.
Bottleneck	19.91 (1.70)	18.33 (3.42)	1.58	Slight Human AMUR advantage	p = .135
Void	21.82 (3.13)	18.63 (4.47)	3.19	Modest Human AMUR advantage	p = .023*
TheFrog	24.36 (3.01)	18.57 (3.43)	5.79	Large Human AMUR advantage	p < .001***
Finger	26.36 (1.91)	19.91 (4.26)	6.45	Large Human AMUR advantage	p < .001***

Table D.11. Average Score per Map for Humans Using the Advanced Maximum Usable Range Strategy Compared to a Computer Agent Using the Path Adjacency Strategy. Sorted by magnitude of human AMUR advantage. * indicates p < .05, *** indicates p < .001.

For the last test, we compared the AMUR group to AI_MUR. A 2 (Human AMUR vs. AI_MUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(4.85, 528.92) = 111.42, MSE = 11.56, p < .001
- a group x map interaction, F(4.85, 528.92) = 14.96, MSE = 11.56, p < .001

- a main effect of group, F(1, 109) = 29.25, MSE = 4.86, p < .001

Across all maps, Human AMUR users (M = 20.81) slightly but reliably outperformed the AI_MUR (M = 19.67), but the group x map interaction indicates that this pattern varied by map. Separate ANOVAs revealed a significant Human AMUR advantage for four maps, but also five maps for which this pattern was reversed, including a significant AI_MUR advantage for three maps.

Next, as we did when looking at the effects of experience, we compared the AMUR group to each baseline AI across all maps at a higher difficulty level.

For AI_Random, a 2 (Human AMUR vs. AI_Random) x 16 (All maps) ANOVA showed:

- a main effect of map, F(9.47, 1032.70) = 118.32, MSE = 20.14, p < .001
- a group x map interaction, F(9.47, 1032.70) = 20.23, MSE = 20.14, p < .001
- a main effect of group, F(1, 109) = 420.27, MSE = 11.60, p < .001

Across all maps, Human AMUR users (M = 13.91) outperformed AI_Random (M = 8.37). The 5.54 point gap represents a 19.8% advantage. Human AMUR significantly outperformed AI_Random on 14 of the 16 maps and underperformed AI_Random on two (Elemental_ish, UpAndDown).

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**AI MUR**			
2in1out	17.36 (2.16)	19.72 (2.01)	-2.36	Modest AI_MUR advantage	p < .001***
Bottleneck	19.91 (1.70)	21.35 (1.29)	-1.44	Slight AI_MUR advantage	p < .001***
NoLeftTurns	13.55 (1.97)	14.60 (1.54)	-1.05	Slight AI_MUR advantage	p = .039*
Pathways	17.00 (4.07)	18.04 (2.45)	-1.04	Slight AI_MUR advantage	Nonsig.
DoTheSplits	13.18 (2.56)	14.16 (1.55)	-0.98	Essentially no difference	p = .068
UpAndDown	26.09 (3.78)	25.61 (2.34)	0.48	Essentially no difference	Nonsig.
TheFrog	24.36 (3.01)	23.15 (2.06)	1.21	Slight Human AMUR advantage	p = .081
Void	21.82 (3.13)	18.52 (0.86)	3.30	Modest Human AMUR advantage	p < .001***
Ladder	23.55 (2.02)	19.76 (2.45)	3.79	Modest Human AMUR advantage	p < .001***
NoSupport	25.73 (3.04)	21.91 (4.94)	3.82	Modest Human AMUR advantage	p = .014*
Finger	26.36 (1.91)	19.53 (1.10)	6.83	Large Human AMUR advantage	p < .001***

Table D.12. Average score per map for humans using the Advanced Maximum Usable Range strategy compared to a computer agent using the Maximum Usable Range strategy. Sorted by magnitude of human AMUR advantage. * indicates p < .05, *** indicates p < .001.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**AI Random**			
Elemental_ish	15.89 (4.13)	18.54 (3.71)	-2.65	Modest AI_Random advantage	p = .028*
UpAndDown	16.98 (5.21)	19.10 (4.30)	-2.12	Modest AI_Random advantage	p = .131
DoTheSplits	5.70 (0.89)	3.27 (2.46)	2.43	Modest Human AMUR advantage	p = .002**
TheFrog	12.14 (2.67)	9.69 (3.60)	2.45	Modest Human AMUR advantage	p = .031*
SnakingPath	17.02 (1.63)	13.72 (4.44)	3.30	Modest Human AMUR advantage	p = .016*
RoundTheTwist	24.89 (2.77)	21.22 (5.41)	3.67	Modest Human AMUR advantage	p = .029*
Void	13.51 (3.19)	9.28 (4.05)	4.23	Large Human AMUR advantage	p = .001**
Pathways	7.84 (1.80)	3.34 (2.69)	4.50	Large Human AMUR advantage	p < .001***
NoLeftTurns	5.89 (1.33)	1.10 (1.93)	4.79	Large Human AMUR advantage	p < .001***
SlimPickings	17.32 (3.47)	11.06 (5.75)	6.26	Large Human AMUR advantage	p < .001***
2in1out	7.48 (0.87)	0.96 (1.71)	6.52	Large Human AMUR advantage	p < .001***
Bottleneck	8.09 (1.06)	1.20 (1.82)	6.89	Large Human AMUR advantage	p < .001***
Switchback	22.45 (2.65)	15.32 (4.36)	7.13	Very large Human AMUR advantage	p < .001***
Ladder	10.81 (1.10)	0.97 (1.75)	9.84	Very large Human AMUR advantage	p < .001***
Finger	19.41 (1.15)	4.10 (4.12)	15.31	Very large Human AMUR advantage	p < .001***
NoSupport	17.14 (3.80)	0.98 (2.27)	16.16	Very large Human AMUR advantage	p < .001***

Table D.13. Average score per map for humans using the Advanced Maximum Usable Range strategy compared to a computer agent placing pieces at random (Advanced Difficulty). Sorted by magnitude of human AMUR advantage. * indicates p < .05, ** p < .01, *** p < .001.

For AI_PathAdjacent, a 2 (Human AMUR vs. AI_PathAdjacent) x 16 (All maps) ANOVA showed:

- a main effect of map, F(10.19, 1110.59) = 94.03, MSE = 16.75, p < .001
- a group x map interaction, F(10.19, 1110.59) = 5.47, MSE = 16.75, p < .001
- a marginal main effect of group, F(1, 109) = 3.23, MSE = 13.54, p = .075

Human AMUR's (M = 13.91) and AI_PathAdjacent's (M = 13.38) performance was quite similar. Human AMUR often showed a slight advantage, and significantly exceeded that of AI_PathAdjacent for three maps (RoundTheTwist, Finger, and Switchback) but AI_PathAdjacent had the numerical advantage for six of the 16 maps and this advantage reached significance for two of the maps (Elemental_ish and Ladder)

For AI_MUR, a 2 (Human AMUR vs. AI_MUR) x 16 (All maps) ANOVA showed:

- a main effect of map, F(6.93, 755.37) = 343.34, MSE = 9.67, p < .001
- a group x map interaction, F(6.93, 755.37) = 15.67, MSE = 9.67, p < .001
- a main effect of group, F(1, 109) = 11.12, MSE = 4.87, p = .001

Human AMUR's (M = 13.91) and AI_MUR's (M = 14.49) overall performance was quite similar. The group x map interaction arose because, although the AI_MUR often exceeded the performance shown by humans using AMUR, in several cases they performed similarly, and in four of the 16 maps the humans using the AMUR strategy significantly outperformed the AI (Finger, Ladder, NoSupport, SlimPickings).

Given that the AMUR strategy maximizes space rather than time despite the problem having a temporal goal, how well does spatial-proxying work? Quite well. Both testing with ceiling effect maps excluded (M = 20.81) and all maps at a higher difficulty to remove the ceiling effect (M = 13.91), it is clear that, in this domain, using space as a proxy for time works significantly better than chance (M = 9.19 / 8.37, gain = 42% / 19.8%).

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**AI PathAdj.**			
Elemental_ish	15.89 (4.13)	19.15 (3.97)	-3.26	Modest AI_PathAdj. advantage	p = .011*
UpAndDown	16.98 (5.21)	19.30 (4.42)	-2.32	Modest AI_PathAdj. advantage	p = .107
Ladder	10.81 (1.10)	12.55 (2.42)	-1.74	Slight AI_PathAdj. advantage	p = .021*
Bottleneck	8.09 (1.06)	9.51 (2.41)	-1.42	Slight AI_PathAdj. advantage	p = .057
2in1out	7.48 (0.87)	8.69 (2.07)	-1.21	Slight AI_PathAdj. advantage	p = .057
DoTheSplits	5.70 (0.89)	6.85 (2.03)	-1.15	Slight AI_PathAdj. advantage	p = .068
Pathways	7.84 (1.80)	8.29 (1.84)	-0.45	Essentially no difference	Nonsig.
NoLeftTurns	5.89 (1.33)	5.96 (2.18)	-0.07	Essentially no difference	Nonsig.
SlimPickings	17.32 (3.47)	16.68 (4.36)	0.64	Essentially no difference	Nonsig.
TheFrog	12.14 (2.67)	11.30 (2.76)	0.84	Essentially no difference	Nonsig.
NoSupport	17.14 (3.80)	15.72 (3.85)	1.42	Slight Human AMUR advantage	Nonsig.
Void	13.51 (3.19)	12.02 (4.60)	1.49	Slight Human AMUR advantage	Nonsig.
SnakingPath	17.02 (1.63)	15.21 (3.83)	1.81	Slight Human AMUR advantage	p = .125
RoundTheTwist	24.89 (2.77)	21.27 (5.23)	3.62	Modest Human AMUR advantage	p = .026*
Finger	19.41 (1.15)	14.90 (2.72)	4.51	Large Human AMUR advantage	p < .001***
Switchback	22.45 (2.65)	16.75 (3.96)	5.70	Large Human AMUR advantage	p < .001***

Table D.14. Average score per map for humans using the Advanced Maximum Usable Range strategy compared to a computer agent using the Path Adjacency strategy (Advanced Difficulty). Sorted by magnitude of human AMUR advantage. * indicates p < .05, *** p < .001.

Map	Mean (SD)		Diff.	Comparison	Significance
	Human AMUR	AI MUR			
Elemental_ish	15.89 (4.13)	22.61 (2.10)	-6.72	Large AI-MUR advantage	p < .001***
Switchback	22.45 (2.65)	25.29 (2.01)	-2.84	Modest AI-MUR advantage	p < .001***
Void	13.51 (3.19)	16.19 (1.63)	-2.68	Modest AI-MUR advantage	p < .001***
RoundTheTwist	24.89 (2.77)	27.36 (1.16)	-2.47	Modest AI-MUR advantage	p < .001***
2in1out	7.48 (0.87)	9.20 (1.02)	-1.72	Slight AI-MUR advantage	p < .001***
Bottleneck	8.09 (1.06)	9.71 (1.26)	-1.62	Slight AI-MUR advantage	p < .001***
SnakingPath	17.02 (1.63)	18.55 (1.75)	-1.53	Slight AI-MUR advantage	p = .007**
UpAndDown	16.98 (5.21)	18.49 (2.84)	-1.51	Slight AI-MUR advantage	p = .132
TheFrog	12.14 (2.67)	12.25 (1.95)	-0.11	Essentially no difference	Nonsig.
Pathways	7.84 (1.80)	7.67 (1.80)	0.17	Essentially no difference	Nonsig.
DoTheSplits	5.70 (0.89)	5.38 (1.04)	0.32	Essentially no difference	Nonsig.
NoLeftTurns	5.89 (1.33)	5.51 (1.50)	0.38	Essentially no difference	Nonsig.
Ladder	10.81 (1.10)	8.95 (1.47)	1.86	Slight Human AMUR advantage	p < .001***
SlimPickings	17.32 (3.47)	15.45 (2.07)	1.87	Slight Human AMUR advantage	p = .01*
Finger	19.41 (1.15)	16.54 (1.47)	2.87	Modest Human AMUR advantage	p < .001***
NoSupport	17.14 (3.80)	12.76 (4.72)	4.38	Large Human AMUR advantage	p = .004*

Table D.15. Average score per map for humans using the Advanced Maximum Usable Range strategy compared to a computer agent using the Maximum Usable Range strategy (Advanced Difficulty). Sorted by magnitude of human AMUR advantage. * indicates p < .05, ** p < .01, *** p < .001.

Condition	Mean			
	Human AMUR	AI_Random	AI_PathAdjacent	AI_MUR
Non-Ceiling Maps	20.81	9.19	19.36	19.67
All Maps	13.91	8.37	13.38	14.49
Advanced Difficulty				

Table D.16. Human use of the Advanced Maximum Usable Range (AMUR) strategy compared to baseline synthetic benchmarks. Average score of the humans using the spatial strategy ADVANCED MAXIMUM USABLE RANGE compared to baseline scores.

An unexpected finding is that humans using the advanced version of the MAXIMUM USABLE RANGE strategy (AMUR) (M = 20.81), which adds the temporal SLOW IN RANGE strategy, do not do appreciably better than the computer agent using the base MAXIMUM USABLE RANGE (MUR) strategy (M = 19.67, gain = 4%). One interpretation is that the temporal SLOW IN RANGE strategy, which potentially sacrifices space to increase time, is no more effective than placing those same towers at locations that maximize space (slowing towers do a small amount of damage so increasing their range potentially increases the amount of damage they do).

An alternate interpretation is that the human's advantage from using the SLOW IN RANGE strategy is matched by the computer agent's ability to accurately measure range, especially compared to human subjects that used satisficing measurement strategies. This would not, however, explain why the human AMUR scores are only slightly better than those of the path adjacent strategy.

A third possibility is skill level. The average score for humans using the AMUR strategy ranged from 19.09 to 25.09, a 6 point (21%) difference. While the average AMUR score is not much higher than the less sophisticated and purely spatial AI_MUR baseline strategy, the better AMUR users significantly outperformed it (gain = 19%).

D.3.2 Spatial Proxying vs. Direct Temporal Manipulation

The spatial AMUR strategy set, which uses space as a proxy for time and attempts to maximize space, performs significantly better than chance. How does it fare relative to the temporal DS strategy set, which treats space and time as independent and attempts to directly maximize time?

A 2 (Human AMUR vs. Human DS) x 11 (non-ceiling maps) mixed factor ANOVA showed:

- a main effect of map, F(10, 140) = 24.69, MSE = 8.17, p < .001
- a group x map interaction, F(10, 140) = 4.03, MSE = 8.17, p < .001
- a main effect of group, F(1, 14) = 34.38, MSE = 9.25, p < .001

The temporal DS strategy (M = 23.71) outperformed the spatial AMUR strategy (M = 20.81) across all non-ceiling maps but the significant map x group interaction shows that the DS group's advantage was more pronounced on some maps than others. Table xx shows the magnitude of the advantage for the direct temporal manipulation strategy (Human DS) over the spatial-proxying strategy (Human AMUR) separately by map.

We also examined the magnitude of the direct temporal manipulation strategy over the spatial proxy strategy for all 16 maps at the simulated increased difficulty level.

A 2 (Human AMUR vs. Human DS) x 16 (non-ceiling maps) mixed factor ANOVA showed:

- a main effect of map, F(15, 210) = 58.20, MSE = 7.39, p < .001
- a group x map interaction, F(15, 210) = 1.91, MSE = 7.39, p = .024
- a main effect of group, F(1, 14) = 12.62, MSE = 20.55, p = .003

The temporal DS strategy (M = 16.08) outperformed the spatial AMUR strategy (M = 13.91) across all 16 maps but the modest but significant map x group interaction shows that the DS group's advantage was more pronounced on some maps than others. The table below shows the magnitude of the advantage for the direct temporal manipulation strategy (Human DS) over the spatial-proxying strategy (Human AMUR) separately by map at the higher difficulty level.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human AMUR	**Human DS**			
NoSupport	25.73 (3.04)	25.00 (2.55)	-0.73	Essentially no difference	Nonsig.
Finger	26.36 (1.91)	26.00 (2.12)	-0.36	Essentially no difference	Nonsig.
Ladder	23.55 (2.02)	23.80 (1.92)	0.25	Essentially no difference	Nonsig.
UpAndDown	26.09 (3.78)	27.40 (0.89)	1.31	Slight Human DiffSlow Advantage	Nonsig.
TheFrog	24.36 (3.01)	25.80 (1.79)	1.44	Slight Human DiffSlow Advantage	Nonsig.
Bottleneck	19.91 (1.70)	22.40 (2.07)	2.49	Modest Human DiffSlow Advantage	p = .023*
2in1out	17.36 (2.16)	20.80 (1.92)	3.44	Modest Human DiffSlow Advantage	p = .009**
Void	21.82 (3.13)	25.60 (3.05)	3.78	Modest Human DiffSlow Advantage	p = .04*
DoTheSplits	13.18 (2.56)	17.20 (4.44)	4.02	Large Human DiffSlow Advantage	p = .036*
NoLeftTurns	13.55 (1.97)	20.40 (6.66)	6.85	Large Human DiffSlow Advantage	p = .006**
Pathways	17.00 (4.07)	26.40 (2.61)	9.40	Very large Human DiffSlow Advantage	p < .001***

Table D.17. Average score per map for humans using the Advanced Maximum Usable Range (AMUR) strategy compared to humans using a Differential Slowing (DS) strategy. Sorted by magnitude of human AMUR advantage. * indicates p < .05, ** p < .01, *** p < .001.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human DS	**Human AMUR**			
Finger	18.60 (2.49)	19.41 (1.15)	-0.81	Essentially no difference	Nonsig.
NoSupport	16.35 (1.44)	17.14 (3.80)	-0.79	Essentially no difference	Nonsig.
Switchback	22.85 (3.97)	22.45 (2.65)	0.40	Essentially no difference	Nonsig.
DoTheSplits	6.25 (2.05)	5.70 (0.89)	0.55	Essentially no difference	Nonsig.
UpAndDown	17.60 (2.71)	16.98 (5.21)	0.62	Essentially no difference	Nonsig.
Ladder	11.85 (1.07)	10.81 (1.10)	1.04	Slight Human DiffSlow Advantage	p = .101
RoundTheTwist	26.35 (1.75)	24.89 (2.77)	1.46	Slight Human DiffSlow Advantage	Nonsig.
SlimPickings	18.95 (4.42)	17.32 (3.47)	1.63	Slight Human DiffSlow Advantage	Nonsig.
2in1out	9.40 (1.72)	7.48 (0.87)	1.92	Slight Human DiffSlow Advantage	p = .009**
TheFrog	14.35 (2.00)	12.14 (2.67)	2.21	Modest Human DiffSlow Advantage	p = .122
Bottleneck	11.65 (1.08)	8.09 (1.06)	3.56	Modest Human DiffSlow Advantage	p < .001***
NoLeftTurns	9.75 (4.60)	5.89 (1.33)	3.86	Modest Human DiffSlow Advantage	p = .019*
SnakingPath	21.00 (6.20)	17.02 (1.63)	3.98	Modest Human DiffSlow Advantage	p = .059
Elemental_ish	20.30 (5.46)	15.89 (4.13)	4.41	Large Human DiffSlow Advantage	p = .094
Void	18.45 (2.43)	13.51 (3.19)	4.94	Large Human DiffSlow Advantage	p = .008**
Pathways	13.60 (2.92)	7.84 (1.80)	5.76	Large Human DiffSlow Advantage	p < .001***

Table D.18. Average score per map for humans using the Advanced Maximum Usable Range (AMUR) strategy compared to humans using a Differential Slowing (DS) strategy (Advanced Difficulty). Sorted by magnitude of human DS advantage. * indicates p < .05, ** p < .01, *** p < .001.

Section D.1.1 Primary Strategies lists five maps that have properties that make them more amenable to temporal strategies than spatial ones. Three of these maps (NoLeftTurns, Pathways and DoTheSplits, all at Level 22 Difficulty) were particularly difficult for subjects who normally used spatial strategies. We now look at each of these maps in turn.

Subjects were divided into two groups, those who used AMUR and those who used DS. Subjects who used neither are not included in these results. Unlike the previous analysis, group membership is based not on the subject's most common strategy but the strategy they used on a specific map. Not all DS users used DS on all map. Subject counts are reported for each map.

A one-way ANOVA on scores for the map NoLeftTurns comparing performance of subjects that used the temporal DS strategy (n = 7) against those who used the spatial AMUR strategy (n = 13) showed a significant effect of group, F(1, 18) = 7.00, MSE = 15.40, p = .016. The DS group (M = 18.71) substantially outperformed the AMUR group (M = 13.85).

A one-way ANOVA on scores for the map Pathways comparing performance of the DS and AMUR groups showed a significant effect of group, F(1, 18) = 29.23, MSE = 14.10, p < .001. The DS group (M = 25.29) substantially outperformed the AMUR group (M = 15.77).

A one-way ANOVA on scores for the map DoTheSplits comparing performance of the DS and AMUR groups showed a significant effect of group, F(1, 18) = 23.47, MSE = 5.54, p < .001. The DS group (M = 19.00) substantially outperformed the AMUR group (M = 12.63).

	NoLeftTurns			Pathways			DoTheSplits		
	Mean	Stdev	N	Mean	Stdev	N	Mean	Stdev	N
DS	18.71	6.157	7	25.29	3.200	7	19.00	2.160	4
AMUR	13.85	2.035	13	15.77	1.003	13	12.63	2.391	16
DS Advantage	17.36%			34.00%			22.75%		

Table D.19. Human AMUR vs. Human DS, by Map.

The (approximate) goal in tower defense is to maximize time. We've shown that, in this domain, space has a strong correlation with time such that maximizing space strongly increases time. How does this compare to attempting to maximize time directly? The temporal strategy DS modestly outperformed the spatial strategy AMUR both when tested on maps without a ceiling effect (10% gain) and when all maps were included and tested on a harder difficulty level (8% gain). The actual gain varied strongly by map, with spatial-proxying working just as well as direct temporal manipulation on some maps and performing much worse on others. There appear to be circumstances under which spatial proxying is appropriate and others where it is not. For an explanation of why some maps were difficult for spatial strategy users and how direct temporal manipulation makes a difference, please see Appendix F Solutions.

	Mean		DS
Condition	**Human AMUR**	**Human DS**	**Advantage**
Non-Ceiling Maps	20.81	23.71	10.4%
All Maps Advanced Difficulty	13.91	16.08	7.8%

Table D.20. Comparison of the two human strategies, the spatial strategy Advanced Maximum Usable Range (AMUR) and the temporal strategy Differential Slowing (DS). Values represent the advantage the agent on the side (rows) has over the agent on top (columns).

D.4 Human vs. Computer

The AIs AI_Random, AI_PathAdjacent and AI_MUR were intended to set baseline performance. The two human strategies tested here, ADVANCED MAXIMUM USABLE RANGE (AMUR) and DIFFERENTIAL SLOWING (DS), performed well above chance (AI_Random). AI_PathAdjacent used a simple heuristic (always place towers next to the path, guaranteeing the tower covers at least a certain number of path tiles). AI_MUR maximized an ordinal spatial property that, while not present in the raw map data, could easily and straight forwardly obtained through brute force. The human use of AMUR outperformed both AI_PathAdjacent and AI_MUR by a small but statistically significant amount when ceiling effect maps were excluded and performed at roughly the same level when they were not. Humans using the DS strategy outperformed all baseline AIs, significantly outperforming them on a certain subset of maps that are not easily solved with spatial strategies.

In this section we look at the performance of two additional computer agents, AI_AMUR and AI_DS, implementing the AMUR and DS strategies respectively. We will refer to these as strategy AIs, as opposed to baseline AIs.

There are two questions of interest regarding these AI. The first regards fidelity. Although the AMUR strategy uses a more advanced notion of usable range and there are nuanced placement optimizations regarding the placement of slowing towers, it is still a primarily spatial strategy and relatively straightforward to understand and implement. In contrast, the temporal DS strategy and the representations it's based on proved surprisingly difficult to translate into computer code. As it is our goal to understand the nature of spatio-temporal reasoning in problem solving, we can only prove that we have correctly understood the relevant concepts by creating an accurate computer simulation of them. We validate this understanding by showing that the AI performs roughly as well as the humans using the same strategies, meaning that it generalizes across a set of strongly dissimilar problems, performs well on those problems the corresponding human strategy performs well on and performs poorly on those problems the corresponding human strategy performs poorly on. If these computer agents perform at the same level as humans on a complex spatio-temporal reasoning task that is not amenable to current AI approaches, we will have achieved something significant.

We measure the performance of these strategies against the performance of the relevant human group. These groups represent an assortment of individuals, each with a different score. The differences between these individuals can be partially explained by differences in how properties are measured and the flexibility to tailor the strategy to subtle aspects of a specific problem. The second question addresses these sources of variability. The computer agents are more accurate in their ability to measure properties such as usable range and path gap but currently lack the flexibility that the humans have to address subtle aspects of a problem. For this reason, it is expected that the computer agents will differ slightly in performance relative to the human groups. We wish to see whether the computer's superior ability to measure quantifiable properties offsets its disadvantage in flexible thinking and ability to perceiving subtle aspects of a problem.

D.4.1 Strategy AIs vs. Baseline AIs

Earlier we measured how human in the two strategy groups (AMUR and DS) compared to three baseline AIs. In this section we compare the two strategy AIs to the baseline AIs.

We begin with the spatial agent AI_AMUR. A 2 (AI_AMUR vs. AI_Random) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, $F(8.54, 1690.82) = 381.44$, $MSE = 11.68$, $p < .001$
- a group x map interaction, $F(8.54, 1690.82) = 231.59$, $MSE = 11.68$, $p < .001$
- a main effect of group, $F(1, 198) = 7358.79$, $MSE = 10.71$, $p < .001$

Across all non-ceiling maps, AI_AMUR (M = 21.16) outperformed AI_Random (M = 9.19). The 11.97/28 point gap represents a 42.8% advantage.

A 2 (AI_AMUR vs. AI_PathAdjacent) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(8.34, 1652.14) = 376.08, MSE = 7.14, p < .001
- a group x map interaction, F(8.34, 1652.14) = 65.68, MSE = 7.14, p < .001
- a main effect of group, F(1, 198) = 268.59, MSE = 6.68, p < .001

Across all non-ceiling maps, AI_AMUR (M = 21.16) outperformed AI_PathAdjacent (M = 19.36). The 1.8 point gap represents a 6.4% advantage.

A 2 (AI_AMUR vs. AI_MUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(5.65, 1117.94) = 545.88, MSE = 6.17, p < .001
- a group x map interaction, F(5.65, 1117.94) = 65.99, MSE = 6.17, p < .001
- a main effect of group, F(1, 198) = 382.93, MSE = 3.20, p < .001

Across all non-ceiling maps, AI_AMUR (M = 21.16) outperformed AI_MUR (M = 19.67). The 1.49 point gap represents a 5.3% advantage.

We next looked at the temporal agent AI_DS. A 2 (AI_DS vs. AI_Random) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(8.35, 1652.80) = 424.48, MSE = 11.36, p < .001
- a group x map interaction, F(8.35, 1652.80) = 172.05, MSE = 11.36, p < .001
- a main effect of group, F(1, 198) = 11914.77, MSE = 10.01, p < .001

Across all non-ceiling maps, AI_DS (M = 23.91) outperformed AI_Random (M = 9.19). The 14.72 point gap represents a 52.6% advantage.

A 2 (AI_DS vs. AI_PathAdjacent) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.88, 1560.69) = 245.38, MSE = 6.94, p < .001
- a group x map interaction, F(7.88, 1560.69) = 152.45, MSE = 6.94, p < .001
- a main effect of group, F(1, 198) = 1913.80, MSE = 5.97, p < .001

Across all non-ceiling maps, AI_DS (M = 23.91) outperformed AI_PathAdjacent (M = 19.36). The 4.55 point gap represents a 16.3% advantage.

A 2 (AI_DS vs. AI_MUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(4.91, 972.41) = 377.38, MSE = 6.10, p < .001
- a group x map interaction, F(4.91, 972.41) = 182.02, MSE = 6.10, p < .001
- a main effect of group, F(1, 198) = 3970.72, MSE = 2.50, p < .001

Across all non-ceiling maps, AI_DS (M = 23.91) outperformed AI_MUR (M = 19.67). The 4.24 point gap represents a 15.1% advantage.

We also compared the two strategy agents against each other. A 2 (AI_AMUR vs. AI_DS) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.71, 1527.16) = 693.19, MSE = 1.41, p < .001
- a group x map interaction, F(7.71, 1527.16) = 691.36, MSE = 1.41, p < .001
- a main effect of group, F(1, 198) = 3765.71, MSE = 1.11, p < .001

Across all non-ceiling maps, AI_DS (M = 23.91) outperformed AI_MUR (M = 21.16). The 2.75 point gap represents a 9.8% advantage.

	AI_Random	**AI_PathAdjacent**	**AI_MUR**	**AI_AMUR**	**AI_DS**
AI_AMUR	42.8%	6.4%	5.3%	-	-9.8%
AI_DS	52.6%	16.3%	15.1%	9.8%	-

Table D.21. Strategy AIs compared to Baseline AIs. Values represent the advantage the agent on the side has over the agent on top.

D.4.2 AIs vs. Domain Experience

Now that we have introduced all five computer agents we return to the topic of domain experience. Each human experience group (domain novice and domain experienced) is compared to each AI.

We begin with the domain novices. A 2 (Novice Human vs. AI_Random) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(8.33, 899.83) = 58.84, MSE = 20.85, p < .001

- a group x map interaction, F(8.33, 899.83) = 27.96, MSE 20.85, p < .001
- a main effect of group, F(1, 108) = 704.90, MSE = 18.33, p < .001

Across all non-ceiling maps, human domain novices (M = 20.56) outperformed AI_Random (M = 9.19). The 11.37 point gap represents a 40.6% advantage.

A 2 (Novice Human vs. AI_PathAdjacent) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.86, 848.97) = 72.84, MSE = 12.75, p < .001
- a group x map interaction, F(7.86, 848.97) = .58, MSE = 12.75, p < .001
- a main effect of group, F(1, 108) = 13.16, MSE = 10.92, p < .001

Across all non-ceiling maps, human domain novices (M = 20.56) outperformed AI_PathAdjacent (M = 19.36). The 1.2 point gap represents a 4.3% advantage.

A 2 (Novice Human vs. AI_MUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(4.89, 528.26) = 113.51, MSE = 11.20, p < .001
- a group x map interaction, F(4.89, 528.26) = 16.92, MSE = 11.20, p < .001
- a main effect of group, F(1, 108) = 17.25, MSE = 4.55, p < .001

Across all non-ceiling maps, human domain novices (M = 20.56) outperformed AI_MUR (M = 19.67). The 0.89 point gap represents a 3.2% advantage.

A 2 (Novice Human vs. AI_AMUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.57, 817.05) = 264.62, MSE = 2.61, p < .001
- a group x map interaction, F(7.57, 817.05) = 80.93, MSE = 2.61, p < .001
- a main effect of group, F(1, 108) = 18.33, MSE = 2.01, p < .001

Across all non-ceiling maps, human domain novices (M = 20.56) underperformed AI_AMUR (M = 21.16). The 0.6 point gap represents a 2.1% advantage for the strategy agent AI_AMUR.

A 2 (Novice Human vs. AI_DS) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(6.23, 672.34) = 345.95, MSE = 1.73, p < .001

- a group x map interaction, F(6.23, 672.34) = 209.99, MSE = 1.73, p < .001
- a main effect of group, F(1, 108) = 1577.68, MSE = 0.72, p < .001

Across all non-ceiling maps, human domain novices (M = 20.56) underperformed AI_DS (M = 23.91). The 3.35 point gap represents a 12% advantage for the strategy agent AI_DS.

Domain novices (M = 20.56) performed better than the baseline AIs and as well as the computer agent using the same strategy but not as well as the strategy agents (AMUR: M = 20.81.

We now look at subjects that had domain experience.

A 2 (Experienced Human vs. AI_Random) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(8.30, 895.89) = 49.19, MSE = 21.32, p < .001
- a group x map interaction, F(8.30, 895.89) = 19.06, MSE = 21.32, p < .001
- a main effect of group, F(1, 108) = 896.55, MSE = 19.95, p < .001

Across all non-ceiling maps, humans with previous domain experience (M = 22.56) outperformed AI_Random (M = 9.19). The 13.37 point gap represents a 47.8% advantage for humans with domain experience.

A 2 (Experienced Human vs. AI_PathAdjacent) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.98, 862.31) = 50.46, MSE = 12.94, p < .001
- a group x map interaction F(7.98, 862.31) = 5.14, MSE = 12.94, p < .001
- a main effect of group, F(1, 108) = 82.01, MSE = 12.55, p < .001

Across all non-ceiling maps, humans with previous domain experience (M = 22.56) outperformed AI_PathAdjacent (M = 19.36). The 3.2 point gap represents an 11.4% advantage for humans with domain experience.

A 2 (Experienced Human vs. AI_MUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(4.97, 536.79) = 74.79, MSE = 11.65, p < .001
- a group x map interaction, F(4.97, 536.79) = 8.65, MSE = 11.65, p < .001
- a main effect of group, F(1, 108) = 135.65, MSE = 6.18, p < .001

Across all non-ceiling maps, humans with previous domain experience (M = 22.56) outperformed AI_MUR (M = 19.67). The 2.89 point gap represents a 10.3% advantage for humans with domain experience.

A 2 (Experienced Human vs. AI_AMUR) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(7.17, 774.03) = 143.77, MSE = 3.19, p < .001
- a group x map interaction, F(7.17, 774.03) = 53.56, MSE = 3.19, p < .001
- a main effect of group, F(1, 108) = 54.17, MSE = 3.63, p < .001

Across all non-ceiling maps, humans with previous domain experience (M = 22.56) outperformed AI_AMUR (M = 21.16). The 1.4 point gap represents a 5.0% advantage for humans with domain experience.

A 2 (Experienced Human vs. AI_DS) x 11 (Non-ceiling maps) ANOVA showed:

- a main effect of map, F(5.89, 635.54) = 193.56, MSE = 2.36, p < .001
- a group x map interaction, F(5.89, 635.54) = 71.36, MSE = 2.36, p < .001
- a main effect of group, F(1, 108) = 77.82, MSE = 2.34, p < .001

Across all non-ceiling maps, humans with previous domain experience (M = 22.56) underperformed AI_DS (M = 23.91). The 1.35 point gap represents a 4.8% advantage in favor of the strategy agent AI_DS.

	AI_Random	AI_PathAdjacent	AI_MUR	AI_AMUR	AI_DS
Novice	40.6%	4.3%	3.2%	-2.1%	-12.0%
Experienced	47.8%	11.4%	10.3%	5.0%	-4.8%

Table D.22. Strategy AIs compared to humans at different experience levels. Values represent the advantage the humans at a given experience level (rows) have over the computer agents (columns).

D.4.3 Strategy AIs vs. Human Strategies

How well do the strategy agents perform relative to humans using the same strategy?

We begin by looking at how the computer implementation of the spatial strategy AMUR did compared human domain novices using the same strategy. A 2 (group: AI_AMUR vs. novice AMUR) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(7.55, 792.63) = 192.11, MSE = 2.46, p < .001
- a group x map interaction, F(7.55, 792.63) = 58.96, MSE = 2.46, p < .001
- a main effect of group, F(1, 105) = 17.68, MSE = 2.03, p < .001

Across all non-ceiling maps, AI_AMUR (M = 21.16) outperformed novice AMUR users (M = 20.46). The 0.7 point gap represents a 2.5% advantage in favor of the strategy agent AI_AMUR.

Including the ceiling effect maps and tested at a higher difficulty level, a 2 (group: AI_AMUR vs. novice AMUR) x 16 (map) mixed factor ANOVA showed:

- a main effect of map, F(9.70, 1018.16) = 339.20, MSE = 3.53, p < .001
- a group x map interaction, F(9.70, 1018.16) = 43.46, MSE = 3.53, p < .001
- a main effect of group, F(1, 105) = 275.67, MSE = 2.50, p < .001

Across all maps, AI_AMUR (M = 16.30) outperformed novice AMUR users (M = 13.74). The 2.56 point gap represents a 9.1% advantage in favor of the strategy agent AI_AMUR. AI_AMUR clearly outperformed the novice AMUR for most maps but humans domain novices had higher scores on three maps (Finger, NoSupport, SlimPickings).

We now switch our attention to AMUR users with domain experience. A 2 (group: AI_AMUR vs. experienced AMUR) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(7.63, 778.01) = 100.87, MSE = 2.42, p < .001
- a group x map interaction, F(7.63, 778.01) = 35.37, MSE = 2.42, p < .001
- no effect of group, F(1, 102) = 1.64, MSE = 1.90, p =.20

Across non-ceiling maps there was no significant difference between AI_AMUR (21.16) and human AMUR users with domain experience (21.43).

Including the ceiling effect maps and tested at a higher difficulty level, a 2 (group: AI_AMUR vs. experienced AMUR) x 16 (map) mixed factor ANOVA showed:

- a main effect of map, F(9.95, 1014.48) = 208.00, MSE = 3.33, p < .001
- a group x map interaction, F(9.95, 1014.48) = 28.07, MSE = 3.33, p < .001
- a main effect of group, F(1, 102) = 90.39, MSE = 2.96, p < .001
- The interaction arose because the AI AMUR clearly outperformed the six experienced AMUR human players for most maps, but not for map 5 (Finger), map 8 (NoSupport), map 11 (SlimPickings), map 15 (UpAndDown)

Across all maps, AI_AMUR (M = 16.30) outperformed human AMUR users with previous domain experience (M = 14.22). The 2.08 point gap represents a 7.4% advantage in favor of the strategy agent AI_AMUR. AI_AMUR clearly outperformed the experienced AMUR group on six of 16 maps and clearly underperformed on four (UpAndDown plus the same three maps domain novices did better on, Finger, NoSupport and SlimPickings).

We now look at the computer's performance using the DS strategy compared to humans using the same strategy. No domain novices used the DS strategy so we only look at DS users with domain experience.

A 2 (group: AI_DS vs. human DS) x 11 (non-ceiling map) mixed factor ANOVA showed:

- a main effect of map, F(6.16, 633.98) = 115.05, MSE = 1.52, p < .001
- a group x map interaction, F(6.16, 633.98) = 39.45, MSE = 1.52, p < .001
- no main effect of group, F(1, 103) = 2.40, MSE = 0.91, p = .125

Across all non-ceiling maps, there was no significant difference between AI_DS (M = 23.91) and human DS users (M = 23.71). As can be seen from the table below, the group x map interaction reflected the finding that the AI_DS significantly outperformed human DS users for four maps, whereas the converse was true for another four maps.

Including the ceiling effect maps and tested at a higher difficulty level, a 2 (group: AI_DS vs. human DS) x 16 (map) mixed factor ANOVA showed:

- a main effect of map, F(9.24, 952.03) = 296.90, MSE = 1.85, p < .001
- a group x map interaction, F(9.24, 952.03) = 44.88, MSE = 1.85, p < .001
- a main effect of group, F(1, 103) = 10.04, MSE = 2.33, p = .002

Across all maps at an advanced difficulty level, there was a slight but significant difference between the AI_DS (M = 15.53) and human DS users (M = 16.08). Follow-up one-way ANOVAs revealed that, at the higher level of difficulty, humans playing the DS strategy outperformed the AI_DS for 7 of the maps.

Map	**Mean (SD)**		**Diff.**	**Comparison**	**Significance**
	Human DS	**AI_DS**			
DoTheSplits	17.20 (4.44)	24.05 (0.80)	-6.85	Large AI_DS advantage	p < .001***
Bottleneck	22.40 (2.07)	24.68 (1.01)	-2.28	Modest AI_DS advantage	p < .001***
Void	25.60 (3.05)	26.84 (0.56)	-1.24	Slight AI_DS advantage	p = .001**
2in1out	20.80 (1.92)	21.81 (0.62)	-1.01	Slight AI_DS advantage	p = .003**
UpAndDown	27.40 (0.89)	27.54 (0.70)	-0.14	Essentially no difference	Nonsig.
Finger	26.00 (2.12)	26.00 (0.00)	0.00	Essentially no difference	Nonsig.
NoLeftTurns	20.40 (6.66)	20.40 (0.49)	0.00	Essentially no difference	Nonsig.
Ladder	23.80 (1.92)	22.11 (1.10)	1.69	Slight Human DS advantage	p = .002**
Pathways	26.40 (2.61)	24.34 (1.22)	2.06	Modest Human DS advantage	p = .001**
TheFrog	25.80 (1.79)	23.66 (0.57)	2.14	Modest Human DS advantage	p < .001***
NoSupport	25.00 (2.55)	21.62 (0.49)	3.38	Modest Human DS advantage	p < .001***

Table D.23. Average score per map for humans using the Differential Slowing (DS) strategy compared to a computer agent using the same strategy. Sorted by magnitude of human DS advantage. * indicates p < .05, ** p < .01, *** p < .001.

Of interest is that the standard deviation over 100 runs of AI_DS on the map NoLeftTurns is zero. There are likely two contributors to this. First, the fewer positions on a map that match a strategy's requirements, the fewer options that agent has, leading to less variance in the agent's solutions. Second, all of an agent's solutions will use positions selected by their strategies. While not common, it is possible that any combination of these positions will lead to the same score.

The DS strategy proved significantly more effective than AMUR on three maps (NoLeftTurns, Pathways, DoTheSplits). We look at each individually.

For the map NoLeftTurns, a one-way ANOVA comparing AI_DS against Experienced Humans who played DS showed no effect of group, F = 2.12, p = .15, with the AI_DS (M = 20.40) and the six Experienced Humans who played DS on this map (M = 19.50) scoring quite similarly, with a slight numerical edge for the AI.

For the map Pathways, a one-way ANOVA comparing AI_DS against Experienced Humans who played DS showed a marginal effect of group, F(1, 105) = 2.96, MSE = 1.98, p = .089, with the seven Experienced Humans who played DS on this map (M = 25.29) scoring marginally higher than the AI (M = 24.34).

For the map DoTheSplits, a one-way ANOVA comparing AI_DS against Experienced Humans who played DS showed a large and significant effect of group, F(1, 102) = 130.36, MSE = 0.75, p < .001, with AI_DS (M = 24.05) markedly outperforming the four Experienced Humans who adopted DS (M = 19.00) on this particular map.

Map	Mean (SD)		Diff.	Comparison	Significance
	Human DS	AI_DS			
DoTheSplits	6.25 (2.05)	12.04 (0.82)	-5.79	Large AI_DS advantage	p < .001***
NoLeftTurns	9.75 (4.60)	13.00 (0.00)	-3.25	Modest AI_DS advantage	p < .001***
TheFrog	14.35 (2.00)	16.20 (1.13)	-1.85	Modest AI_DS advantage	p = .001**
UpAndDown	17.60 (2.71)	18.16 (0.96)	-0.56	Essentially no difference	Nonsig.
Ladder	11.85 (1.07)	11.79 (0.96)	0.06	Essentially no difference	Nonsig.
Bottleneck	11.65 (1.08)	11.31 (0.76)	0.34	Essentially no difference	Nonsig.
Elemental_ish	20.30 (5.46)	19.90 (0.94)	0.40	Essentially no difference	Nonsig.
Pathways	13.60 (2.92)	13.08 (1.38)	0.52	Essentially no difference	Nonsig.
Void	18.45 (2.43)	17.80 (0.96)	0.65	Essentially no difference	Nonsig.
NoSupport	16.35 (1.44)	14.32 (0.47)	2.03	Modest Human DS advantage	p < .001***
SlimPickings	18.95 (4.42)	16.44 (0.82)	2.51	Modest Human DS advantage	p < .001***
RoundTheTwist	26.35 (1.75)	23.72 (0.78)	2.63	Modest Human DS advantage	p < .001***
Finger	18.60 (2.49)	15.28 (0.96)	3.32	Modest Human DS advantage	p < .001***
SnakingPath	21.00 (6.20)	17.54 (0.98)	3.46	Modest Human DS advantage	p < .001***
2in1out	12.99 (0.10)	9.40 (1.72)	3.59	Modest Human DS advantage	p < .001***
Switchback	22.85 (3.97)	14.87 (1.33)	7.98	Large Human DS advantage	p < .001***

Table D.24. Average score per map for humans using the Differential Slowing (DS) strategy compared to a computer agent using the same strategy (Advanced Difficulty). Sorted by magnitude of human DS advantage. ** indicates p < .01, *** p < .001.

	NoLeftTurns			Pathways			DoTheSplits		
	Mean	**Stdev**	**N**	**Mean**	**Stdev**	**N**	**Mean**	**Stdev**	**N**
AI_DS	20.40	0.492	100	24.34	1.216	100	24.05	0.796	100
Human	19.50	6.348	6	25.29	3.200	7	19.00	2.160	4

Table D.25. AI_DS vs. human Differential Slowing (DS) users, by map.

	AI_AMUR	**AI_DS**
Novice AMUR	-2.5%	-12.3%
Experienced AMUR	non-significant	-8.9%
DS	9.1%	non-significant

Table D.26. Humans vs. computer agents, both within and across strategies. Values represent the advantage the human strategy (rows) has over the computer agent (rows).

In conclusion, within strategies, neither the computer nor humans with domain experience had a large advantage over the other when both used the same strategy. Across strategies, agents that used DS, a strategy that treats time and space as independent, outperform agents that used AMUR, a strategy that uses space as a proxy for time.

Appendix E

Computer API

E.1 Key Classes

What follows is a list of 32 classes used to support spatio-temporal reasoning.

E.1.1 Coordinate System

Spatial reasoning requires thinking about data in multiple qualitatively different and often concurrent coordinate systems.

- `WorldPoint`. A position in the real world, or in this case, the game, which is a continuous space. Creep position and range area are measured in world coordinates.
- `GridPoint`. The real world is discretized to a set of tiles. Tower position is measured in grid coordinates.
- `LocalPoint`. Local positions are used for object-centric coordinates. For example, to say that we should place a slowing tower near an offense tower, the absolute coordinates are irrelevant, the important information is the location of items relative to the offense tower.

E.1.2 Spatial Data

Map classes represent spatial and some temporal information, storing it in a way that makes it easy for the SpatialReasoning classes to function.

- `MapModel`. Represents the raw map data. This includes the structural information (MapGeometry), the identity of each tile (tower, wall, path and floor) and path information (a list of PathSegments). Supports queries on raw data such as "get all unoccupied wall tiles".
- `MapGeometry`. Represents the raw geometry of a map as represented by tiles (GridPoints).
- **List of Path Node Positions**. Paths are represented as an ordered list of positions. Creeps move from the position of one path node to the next, continuing until the last path node. In GopherTD there are two lists, one for each path.

E.1.3 Spatial Meta-Data

- `AnnotatedMapModel`. The `AnnotatedMapModel` contains metadata information about the map. This information makes it easy for SAQS to perform affordance queries. Properties include statistical information (path length, number of corners, number of walls, number of towers, etc.), position queries (e.g., unoccupied wall cells where a tower will cover a specified path cell, unoccupied wall cells where a tower's first attack window will end before a given path position), relationship data (e.g., the distance of a given node from the closest object in a specified set of objects) and tower data (e.g., position of all cleanup towers for path 1). A full list of the `AnnotatedMapModel` methods is given later in this section.
- `MapCellMetaData`. `MapCellMetaData` tracks additional data for each tile on the map. This data includes how whether it's part of a path (and if so, what its

temporal position in the path is), is a corner, is a range entry point, is a range exit point, how many slowing towers cover it and how many offense towers cover it.

- `LocalNeighborhood`. When an individual object performs spatial reasoning about itself, it thinks in terms of its local neighborhood. For a tower, the local neighborhood includes all tiles (`GridPoint`s) in its range. The local neighborhood also includes the direction of any paths that pass through it. Multiple neighborhoods can be tested for intersection. These three properties (tiles in range, path direction, intersection) allow for the representation of relationships such as "slowing tower X's range overlaps the front part of offense tower Y's range" and strategies such as "have slow towers cover every path tile in tower Y's range" (requires the query "which path cells in tower Y's range are not covered by a slowing tower?" and "which positions for slowing tower X would result in position Z being covered?").
- `GroupNeighborhood`. A local neighborhood represents the area around a single object. A group neighborhood represents the area around a specified set of towers. This allows for the representation of kill zones, functional groups and chains and strategies such as "slow creeps as they end and exit a kill zone".

E.1.4 Specialized Spatial Meta-Data

- `MapGeometrySubsection`. Represents an arbitrarily sized portion of `MapGeometry`. Supports rotation, which is used by `SAQS` to perform rotation-invariant pattern matching.
- **Patterns**
    - o `DesiredGridCellType`. Used by `MapGeometryPattern` to specify which types of tiles it's looking for. Values: `DontCare`, `EmptyWall`, `OccupiedWall`, `AnyPath`, `PathA`, `PathB`. Path can be represented generically or specifically. Patterns that specify specific paths can be used to assure equal path coverage (i.e., one instance of a pattern is

applied to one path, another instance is applied to the other; this is useful for maps with split paths).

  - `MapGeometryPattern`. A size-sensitive but rotation-invariant spatial pattern. A 2D array of `DesiredGridCellType`.
  - `MapGeometryPatternFactory`. Returns known `MapGeometryPattern`s. This is an inelegant software solution (a better approach would be data driven) and does not support learning new patterns from inference. This was implemented this way for this research as it was a fast, functional way to accomplish the goals of this project.

- **Influence Maps**
  - `InfluenceMap`. An influence map. Tracks the value of an arbitrary property at each tile on the map. Allows queries for highest/lowest values on the map. Extremely useful for many spatial reasoning tasks but only used for `DistanceMap`s in this project.
  - `DistanceMap`. A type of influence map. Each tile records the distance from that tile to the nearest object, where the set of objects to track are specified in the query. Example: Return all unoccupied wall tiles that are near the offense towers, sorted by which are closest.
- `ValueTransitionPoint`. Value transition points are points along a path where some arbitrary property changes value. Used to mark the boundaries of a region. Example: Calculate the MAXIMUM USABLE RANGE of a level 3 Red3 tower at the best position on the map. Move along the path and calculate the MAXIMUM USABLE RANGE of that tower at any spot behind that point. Keep moving this position, marking each spot where the value changes. This spot is a `ValueTransitionSpot`. Useful for determining where to divide a map between functional groups that cannot be in the same region.

E.1.5 Temporal Data

- `Direction`. Allows spatial reasoning to be done on absolute (north, south, etc.) or relative (right, left) directions.
- **Path Data**
  - `Path`. Represents a full path. Represented as a list of `PathNode`s. Supports many metadata queries such as number of corners, number of turns in a given direction, whether a given path node is spatially contiguous with a node on the other path, whether a given path node is both spatially and temporally contiguous with a node on the other path. These properties make it easier for SAQS to calculate properties such as path synchronization gap, determine whether a map is a split path map and classify maps in strategy-specified ways (e.g., dynamic path gap vs. persistent path gap for the DIFFERENTIAL SLOWING strategies).
  - `PathSegment`. A straight line in a path. Includes its start and end positions, length, direction, the path it belongs to and its order in the path. Segments can also tell whether they contain a specified tile and are in range of a specified tower.
  - `PathNode`. Metadata for path tiles. Includes position, direction, whether it's a corner, the global temporal ID (i.e., its position in the overall path) and, when in a `LocalNeighborhood`, the local temporal ID.
- **Attack Windows**
  - `AttackWindow`. A spatially contiguous section of path(s) that is inside a tower's attack range. A tower can have more than one attack window. Also tracks temporal data for the purposes of calculating path synchronization gaps. This class was intended to also represent temporal attack windows to support the TEMPORAL ATTACK WINDOW SEPARATION strategy but this strategy did not end up being implemented.
  - `AttackWindows`. A set of attack windows for a given tower.

E.1.6 Spatio-Temporal Reasoning

- `MapAnalyzer`. `MapAnalyzer` is a factory class that creates map analysis information. It has two functions. First, it takes the raw map geometry (`MapModel`) and returns an object that supports SAQS' spatio-temporal queries (`AnnotatedMapModel`). Second, `MapAnalyzer` creates `LocalNeighborhood` objects for specified towers.
- `SAQS`. The Spatial Affordance Query System is the heart of the spatio-temporal reasoning system described in this work. It provides the problem solver a means of conducting affordance queries. The problem solver can ask for positions that match a given absolute or relative property, positions that maximize or are above a specified threshold for an ordinal property, spatial configurations that match a pattern, etc. A full list of the SAQS methods is given later in this section.
- `MapPropertyAbsolute`. Properties of a map that are not dependent on a provided parameter. Examples include corners, exterior walls and where two adjacent paths split. Strategies can be represented as an operation on these properties such as "place towers near where two separate paths join together".
- `MapPropertyOrdinal`. Properties of a map that can be ranked (e.g., number of path cells in range). Strategies can be represented as an operation on these properties such as "place towers near where the path synchronization gap is the highest".
- `PlacementAnchor`. Properties that can serve as placement anchors. Strategies can be represented as an operation on these properties such as "place towers on U-turns", "place towers between a pair of attack windows" and "place towers in a group of cleanup towers".
- `PlacementRelationship`. A relationship between a placement target and placement anchor. Values include RCC8 relationships such as inside (RCC8: NTPP) and non-RCC relationships such as near, between, in a chain, in the same area and spread across the map so that the entire path is covered.

- `PlacementTarget`. Conceptual objects that can be placed on the map. In this domain, the problem solver can only physically place a tower but in the reasoning process the problem solver thinks in terms of placing a tower's range, the tower's range entrance, a slowing effect, a pair of attack windows, etc.
- `SAQSPatternMatch`. Represents a match found for a specified pattern (MapGeometryPattern) on a given map. Indicates position and rotation.
- `EvaluationOfRegionSplitForDifferentialSlowing`. Given a set of properties, the value transition points for those properties and other criteria such as the minimum number of tiles in a region and maximum acceptable loss for a property, this class determines how to split a map into regions. This specific class determines how to divide the map up for the DIFFERENTIAL SLOWING strategy, which tries to place eight slowing towers before the kill zone without taking up so much room for the slowing towers that the kill zone loses too many of its top positions to place a set of offense towers.

E.2 AnnotatedMapModel API

The following methods represent the public API implemented for the `AnnotatedMapModel` class. The `MapAnalyzer` class takes the raw map geometry and creates an annotated version of it, adding information such as "this position is a corner" and "this path tile is X steps away from the map entrance". Each method was written in response to a need from one of the developed agents. These methods are normally called by the `SAQS` (Spatial Affordances Query System) class, not the agent. They are presented here to give the reader a sense of the types of information required to implement a spatio-temporal reasoning agent as well as the number of queries needed.

```
addTower(TowerLog tower, int x, int y, int groupID) : void
canPlaceTower(int x, int y) : bool
getCellMetaData() : List<MapCellMetaData>
getCellsOfType(GridCellType type) : List<MapCellMetaData>
getCellType(int x, int y) : GridCellType
getHeight() : int
```

```
getNumberOfCorners() : int
getNumberOfFloorTiles() : int
getNumberOfFreeWallTiles() : int
getNumberOfTilesOfType(GridCellType type) : int
getNumberOfTowers() : int
getNumberOfVirtualPathTiles() : int
getNumberOfWallTiles() : int
getPathNode(GridPoint position) : PathNode
getPositionOfAttackWindowsBeforePathCell(
   TowerModel tower, GridPoint pathPosition, int pathID)
   : List<GridPoint>
getPositionOfAttackWindowsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfAttackWindowsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID,
   List<GridPoint> candidates) : List<GridPoint>
getPositionOfCellsOfType(GridCellType type) : List<GridPoint>
getPositionOfFreeAttackWindowsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfFreeWallCells() : List<GridPoint>
getPositionOfFreeWallsAfterPathCell(TowerModel tower,
   GridPoint pathPosition, int pathID) : List<GridPoint>
getPositionOfFreeWallsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfFreeWallsCoveringPath(TowerModel tower, int pathID)
   : List<GridPoint>
getPositionOFreefWallsAfterPathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfWallsAfterPathCell(TowerModel tower,
   GridPoint pathPosition, int pathID) : List<GridPoint>
getPositionOfWallsAfterPathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfWallsAfterPathCell(TowerModel tower,
   int globalStepNumber, int pathID,
   List<GridPoint> candidates) : List<GridPoint>
getPositionOfWallsBeforePathCell(TowerModel tower,
   GridPoint pathPosition, int pathID) : List<GridPoint>
getPositionOfWallsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID) : List<GridPoint>
getPositionOfWallsBeforePathCell(TowerModel tower,
   int globalStepNumber, int pathID,
   List<GridPoint> candidates) : List<GridPoint>
getPositionOfWallsCoveringPath(TowerModel tower, int pathID)
   : List<GridPoint>
getPositionsInRange(GridPoint cellPosition, float rangeInWorldUnits)
   : List<GridPoint>
getPositionsOfPhysicalPathCellsInRange(GridPoint cellPosition,
   float rangeInWorldUnits) : List<GridPoint>
getTowerDistanceMap(int towerGroupID) : DistanceMap
getWidth() : int
isInRange(TowerModel tower, GridPoint target) : bool
isInRange(WorldPoint center, WorldPoint target, float range)
   : bool
```

```
isOnPath(int x, int y) : bool
```

E.3 SAQS API

The following methods represent the public API implemented for the SAQS (Spatial Affordance Query System) class. All methods were written in response to a need from one of the developed agents. The methods are presented here to give the reader a sense of the types of information required to implement a spatio-temporal reasoning agent as well as the number of queries needed.

```
getDifference(List<GridPoint> a, List<GridPoint> b)
   : List<GridPoint>
getLocalNeighborhood(TowerModel tower, AnnotatedMapModel map)
   : LocalNeighborhood
getLocalNeighborhoodAt(TowerModel tower, AnnotatedMapModel map,
   GridPoint position) : LocalNeighborhood
getMaximumUsableRange(TowerModel tower, AnnotatedMapModel map)
   : int
getMaximumUsableRange(TowerModel tower, AnnotatedMapModel map,
   List<GridPoint> candidates) : int
getNumberOfDifferentialSlowingPositions(int pathID,
   TowerModel tower, AnnotatedMapModel map) : int
getNumberOfDifferentialSlowingPositions(int pathID,
   TowerModel tower, AnnotatedMapModel map,
   List<GridPoint> candidates) : int
getPathAdjacentWallCells(AnnotatedMapModel map,
   List<GridPoint> candidates) : List<GridPoint>
getPathCoverageRatioFor(TowerModel tower, AnnotatedMapModel map)
   : float
getPathCoverageRatioFor(TowerModel tower, AnnotatedMapModel map,
   GridPoint position) : float
getPathGapFor(TowerModel tower, AnnotatedMapModel map) : int
getPatternMatches(MapModel map, MapGeometryPattern pattern)
   : List<SAQSPatternMatch>
getPatternMatchesRotationInvariant(MapModel map,
   MapGeometryPattern pattern) : List<SAQSPatternMatch>
getPositionsByAbsoluteProperty(MapPropertyAbsolute property,
   MapModel mapGeometry) : List<GridPoint>
getPositionsByAbsoluteProperty(MapPropertyAbsolute property,
   AnnotatedMapModel map) : List<GridPoint>
getPositionsByAbsoluteProperty(MapPropertyAbsolute property,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : List<GridPoint>
getPositionsByDualPathCellsInRangeAtLeast(TowerModel tower,
   int threshold, AnnotatedMapModel map,
   List<GridPoint> candidates) : List<GridPoint>
```

```
getPositionsByDualPathRelativePropertyValueAtLeast(
   TowerModel tower, MapPropertyOrdinal property,
   int threshold, AnnotatedMapModel map) : List<GridPoint>
getPositionsByDualPathRelativePropertyValueAtLeast(
   TowerModel tower, MapPropertyOrdinal property, int value,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : List<GridPoint>
getPositionsByMaxPathCellsInRange(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : List<GridPoint>
getPositionsByMaxPathSyncGap(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : List<GridPoint>
getPositionsByMaxPhysicalPathCellsInRange(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : List<GridPoint>
getPositionsByPathCellsInRangeAtLeast(TowerModel tower,
   int threshold, AnnotatedMapModel map,
   List<GridPoint> candidates) : List<GridPoint>
getPositionsByPathSyncGapAtLeast(TowerModel tower,
   int threshold, AnnotatedMapModel map,
   List<GridPoint> candidates) : List<GridPoint>
getPositionsByPhysicalPathCellsInRangeAtLeast(TowerModel tower,
   int threshold, AnnotatedMapModel map,
   List<GridPoint> candidates) : List<GridPoint>
getPositionsByRelativeMaxValue(TowerModel tower,
   MapPropertyOrdinal property, AnnotatedMapModel map)
   : List<GridPoint>
getPositionsByRelativeMaxValue(TowerModel tower,
   MapPropertyOrdinal property, AnnotatedMapModelmap,
   List<GridPoint>candidates) : List<GridPoint>
getPositionsByRelativePropertyValueAtLeast(TowerModel tower,
   MapPropertyOrdinal property, int threshold,
   AnnotatedMapModelmap) : List<GridPoint>
getPositionsByRelativePropertyValueAtLeast(TowerModel tower,
   MapPropertyOrdinal property, int value,
   AnnotatedMapModelmap, List<GridPoint>candidates)
   : List<GridPoint>
getPositionsForSlowingOneLineAttackWindow(TowerModel tower,
   AnnotatedMapModel map, int pathID) : List<GridPoint>
getPositionsForSlowingOneLineAttackWindow(TowerModel tower,
   AnnotatedMapModel map, int pathID,
   List<GridPoint> candidates) : List<GridPoint>
getPositionsForSlowingOneLineAttackWindow(TowerModel tower,
   AnnotatedMapModel map, int pathID, int windowID,
   List<GridPoint> candidates) : List<GridPoint>
getPositionsForSlowingOneLineFullRange(TowerModel tower,
   AnnotatedMapModel map, int pathID) : List<GridPoint>
getPositionsForSlowingOneLineFullRange(TowerModel tower,
   AnnotatedMapModel map, int pathID,
   List<GridPoint> candidates) : List<GridPoint>
getRegionTransitionPointsForDS(int pathID, TowerModel tower,
   AnnotatedMapModel map) : List<ValueTransitionPoint>
```

```
getRegionTransitionPointsForMUR(int pathID, TowerModel tower,
   AnnotatedMapModel map) : List<ValueTransitionPoint>
getScoreForUsableRangeAndPathGapFor(TowerModel tower,
   AnnotatedMapModel map) : int
getScoreForUsableRangeAndPathGapFor(TowerModel tower,
   GridPoint position, AnnotatedMapModel map) : int
getScoresForPathCellsInRange(TowerModel tower,
   AnnotatedMapModel map) : PositionScores
getScoresForPathCellsInRange(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : PositionScores
getScoresForPathCoverageRatio(TowerModel tower,
   AnnotatedMapModel map) : PositionScores
getScoresForPathCoverageRatio(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : PositionScores
getScoresForPathSyncGap(TowerModel tower, AnnotatedMapModel map)
   : PositionScores
getScoresForPathSyncGap(TowerModel tower, AnnotatedMapModel map,
   List<GridPoint> candidates) : PositionScores
getScoresForPhysicalPathCellsInRange(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : PositionScores
getScoresForUsableRangeAndPathGap(TowerModel tower,
   AnnotatedMapModel map) : PositionScores
getScoresForUsableRangeAndPathGap(TowerModel tower,
   AnnotatedMapModel map, List<GridPoint> candidates)
   : PositionScores
getUsableRangeFor(TowerModel tower, AnnotatedMapModel map) : int
getUsableRangeFor(TowerModel tower, AnnotatedMapModel map,
   GridPoint position) : int
isAMatch(DesiredGridCellType lookingFor, GridCellType found)
   : bool
isAMatch(MapGeometryPattern pattern,
   MapGeometrySubsection subsection) : bool
```

Appendix F

Solutions

A common question for both subjects and people learning about this work is whether there is a perfect solution for every map. Maps vary in difficulty and we originally believed that some maps were not only unwinnable but that it was impossible to achieve even moderately high scores on them. While some of the maps were not solved by any subject, by merging insights gleaned from several subjects we were able to discover solutions to every map. Of the 16 maps, subjects found solutions to 12 maps and the GopherTD team found solutions to the remaining four. We present these solutions here. Where possible, we show solutions using the towers subjects were required to use in phase three. Where numerous qualitatively different solutions exist, we show several solutions as a means of illustrating the breadth of successful options.

This appendix is organized into three sections, divided by map difficulty. Table Table F.1 shows the average score for each map by experiment phase. The primary difference between the phases is tower selection. Subjects had 110 tower types to choose from in phase one. We removed the least commonly used and least successful towers, leaving 40 tower types in phase two. In phase three, subjects were not allowed to choose which towers to use. Average scores rose each time we constrained the number of tower types available.

	Experiment			**Mean**	**Category**	**Perfect Scores**	
	1	**2**	**3**				
RoundTheTwist	26.15	27.52	28.00	27.22	Easy	51/58	88%
Switchback	24.31	25.04	28.00	25.78	Easy	41/58	71%
SlimPickings	-	22.84	27.10	24.97	Medium	28/45	62%
UpAndDown	22.00	23.64	26.75	24.13	Medium	31/58	53%
SnakingPath	23.54	24.28	27.50	25.11	Medium	27/58	47%
Elemental-ish	22.38	22.96	27.20	24.18	Medium	27/58	47%
Finger	-	-	26.15	26.15	Medium	8/20	40%
TheFrog	-	19.88	25.25	22.57	Medium	11/45	24%
NoSupport	-	-	25.65	25.65	Medium	3/20	15%
Void	14.23	19.96	22.80	19.00	Hard	7/58	12%
Pathways	-	15.60	19.10	17.35	Hard	5/45	11%
NoLeftTurns	-	-	15.55	15.55	Hard	1/20	5%
Ladder	-	18.44	23.70	21.07	Hard	0/45	0%
Bottleneck	12.46	15.20	20.35	16.00	Hard	0/58	0%
2in1out	12.15	14.76	17.95	14.95	Hard	0/58	0%
DoTheSplits	-	-	13.90	13.90	Hard	0/20	0%

Table F.1. Number of Subjects Earning a Perfect Score per Map. Sorted by number of perfect scores then mean. Maximum score is 28.

The maps are ranked by difficulty as determined in phase three. This ranking is slightly different from the one produced by ranking on the composite average but is a better reflection of map difficulty as it factors out the influence of tower selection strategies. The assigned difficulty rating of each map is based a mixture of average subject scores, the number of subjects who earned a perfect score on the map and the category it was assigned to in the original Vector TD game. Based on our organization, there are two easy problems, seven moderately difficult problems and seven hard problems.

Table F.1 also shows the number of subjects who earned a perfect score for each map. Where no subject achieved a perfect score, we looked at the solutions that had achieved the highest score, used the knowledge gained from the analysis of strategies and features to enhance these and then repeatedly generated solutions until a perfect score was earned. Unlike the subjects, we were allowed repeated attempts to solve each problem. In all four cases, it took us six attempts to create a solution that earned a perfect score. We attempted to use the same 24 towers used in phase three but had to make small changes in two of the four solutions (in both situations, we used fewer blue slowing towers so that we could add cleanup areas that used brown freezing towers).

This appendix is divided into three sections based on problem difficulty. For each problem, a table similar to the following is given:

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	19.88	25.25	16.60	18.57
Range	-	7-28	20-28	9-25	8-26
Standard Deviation	-	5.89	2.63	3.60	3.43
Perfect Scores	-	3/25	5/20	-	-

Table F.2. Scores per Experiment.

The score mean, range, standard deviation and the number of people who got a perfect score (28) are given for each phase. A "-" means that map was not used in that phase. Scores are also given for two baseline AIs. RANDOM places towers randomly across the map, PATH ADJACENT places towers randomly along the path. means they use the same towers as human subjects used in phase three (four level 10 Green3 offense towers, 20 level 1 Blue1 slowing towers). These are the towers used by many of the highest scoring subjects, giving the baseline algorithms a strong advantage over human subjects in phases one and two.

F.1 Easy Maps

Almost all subjects achieved high or perfect scores on the easy difficulty maps. We show a few solutions for each map without elaborate comment.

Round the Twist	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	26.15	27.52	28.00
Range	11-28	23-28	28-28
Standard Deviation	4.95	1.23	0.00

Switchback	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	24.31	25.04	28.00
Range	10-28	13-28	28-28
Standard Deviation	6.06	4.70	0.00

Table F.3. Scores for Easy Maps.

The majority of subjects earned perfect scores on the easy maps. In the following section we show six example solutions from Experiment 2. All six were required to use the same towers but chose different positions. Key items to look for include distribution density (are towers near each other, closely packed or widely distributed), usable range (U-turns and double-sided corners have more than positions in the middle of a wall) and arrangement of blue slowing towers (is there a pattern? Are they around the offense towers or in line with them? Are they closely packed or spaced out?).

The same set of subjects (four domain experienced, two domain novices) are shown for each problem in the same order to make comparison across problems easier.

F.1.1 Round the Twist

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	26.15	27.52	28.00	27.29	26.70
Range	11-28	23-28	28-28	18-28	16-28
Standard Deviation	4.95	1.23	0.00	1.87	2.39
Perfect Scores	11/13	21/25	20/20	-	-

Table F.4. Scores for Round the Twist, with Benchmarks.

In the map Round the Twist, all wall positions other than the external walls are passed twice by each line of creeps. Walls are skinny so that even towers with a small range can reach both sides, giving the towers at least two attack windows (i.e., the entire line of creeps pass by the tower and then later pass it again). Because there are few bad positions on this map, even the random placement strategy earns a near perfect score.

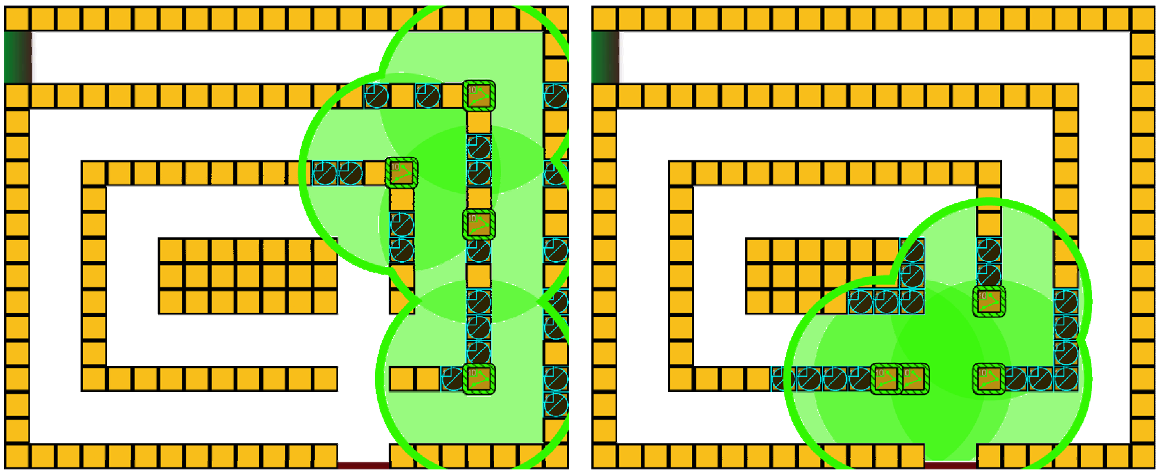


Figure F.1. Perfect Solutions to Round the Twist. LEFT: 3E100. RIGHT: 3E0200.

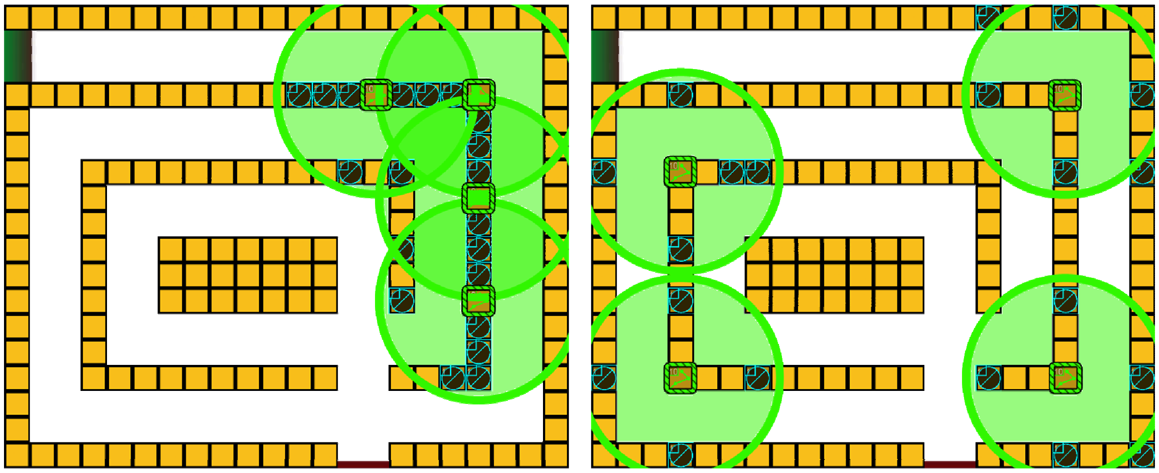


Figure F.2. Perfect Solutions to Round the Twist. LEFT: 3E0300. RIGHT: 3E0400.

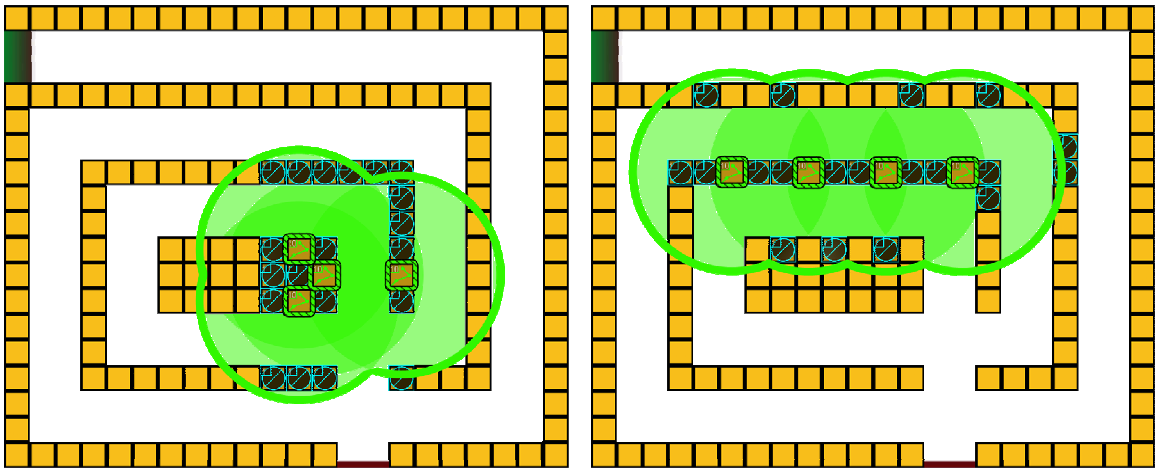


Figure F.3. Perfect Solutions to Round the Twist. LEFT: 3N0400. RIGHT: 3N0900.

F.1.2 Switchback

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	24.31	25.04	28.00	24.94	26.43
Range	10-28	13-28	28-28	12-28	19-28
Standard Deviation	6.06	4.70	0.00	3.13	2.23
Perfect Scores	8/13	11/25	20/20	-	-

Table F.5. Scores for Switchback, with Benchmarks.

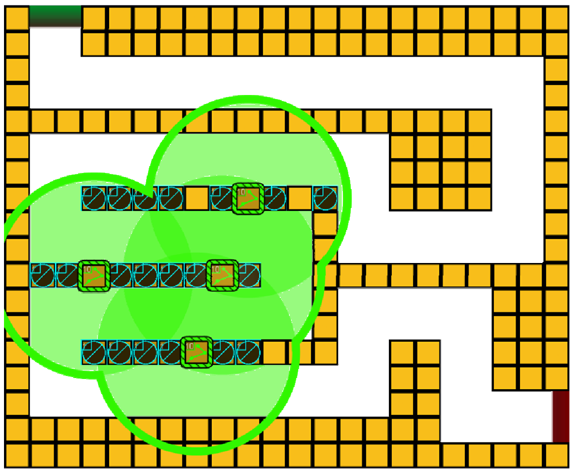
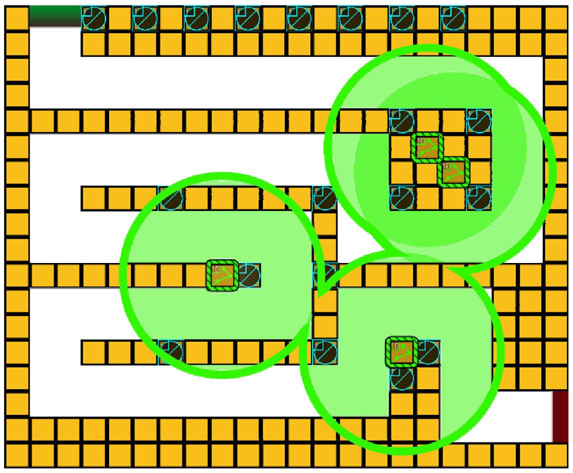



Figure F.4. Perfect Solutions to Switchback. LEFT: 3E100. RIGHT: 3E0200.

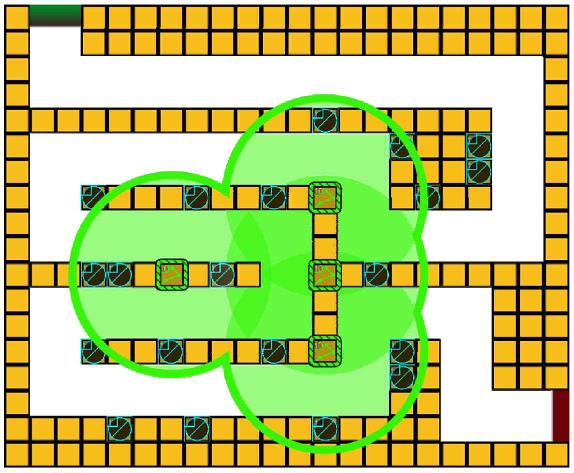
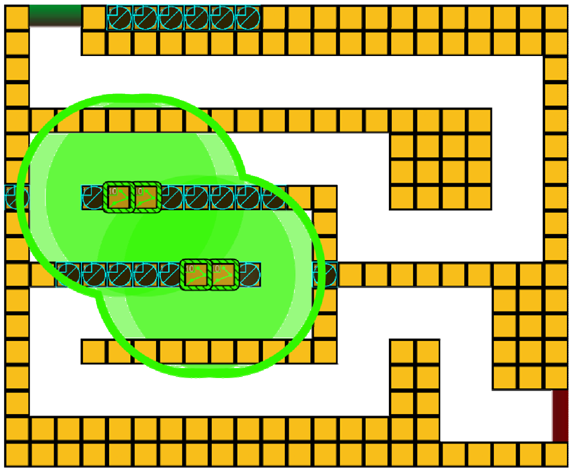



Figure F.5. Perfect Solutions to Switchback. LEFT: 3E0300. RIGHT: 3E0400.

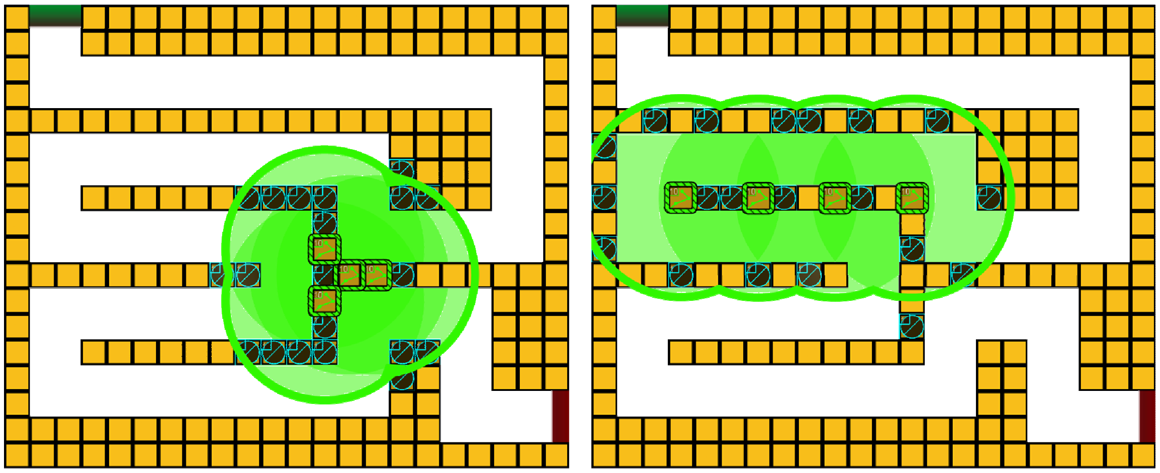


Figure F.6. Perfect Solutions to Switchback. LEFT: 3N0400. RIGHT: 3N0900.

F.2 Moderate Difficulty Maps

	Elemental-ish			**Finger**		
	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	22.38	22.96	27.20	-	-	26.15
Range	4-28	10-28	21-28	-	-	22-28
Standard Deviation	7.24	4.87	1.64	-	-	1.90

	No Support			**Snaking Path**		
	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	-	-	25.65	23.54	24.28	27.50
Range	-	-	17-28	7-28	12-28	26-28
Standard Deviation	-	-	2.54	7.36	5.05	0.69

	Slim Pickings			**The Frog**		
	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	-	22.84	27.10	-	19.88	25.25
Range	-	11-28	17-28	-	7-28	20-28
Standard Deviation	-	5.78	2.49	-	5.89	2.63

	Up and Down			**Void**		
	Exp. 1	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
Average Score	22.00	23.64	26.75	14.23	19.96	22.80
Range	8-28	13-28	15-28	3-28	9-28	18-28
Standard Deviation	7.76	5.27	2.88	9.15	5.33	3.25

Table F.6. Scores for Normal Maps.

For moderately difficult problems, we show one solution and two additional views showing the range covered by the primary offense towers and a set of slowing towers.

For many of these problems, alternative solutions are shown to give a sense of the different types of approaches that are possible and still achieve a perfect score. To encourage this view, solutions on each problem are chosen from different subjects.

F.2.1 Elemental-ish

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	22.38	22.96	27.20	26.84	26.77
Range	4-28	10-28	21-28	21-28	21-28
Standard Deviation	7.24	4.87	1.64	1.61	1.72
Perfect Scores	4/13	9/25	13/20	-	-

Table F.7. Scores for Elemental-ish, with Benchmarks.

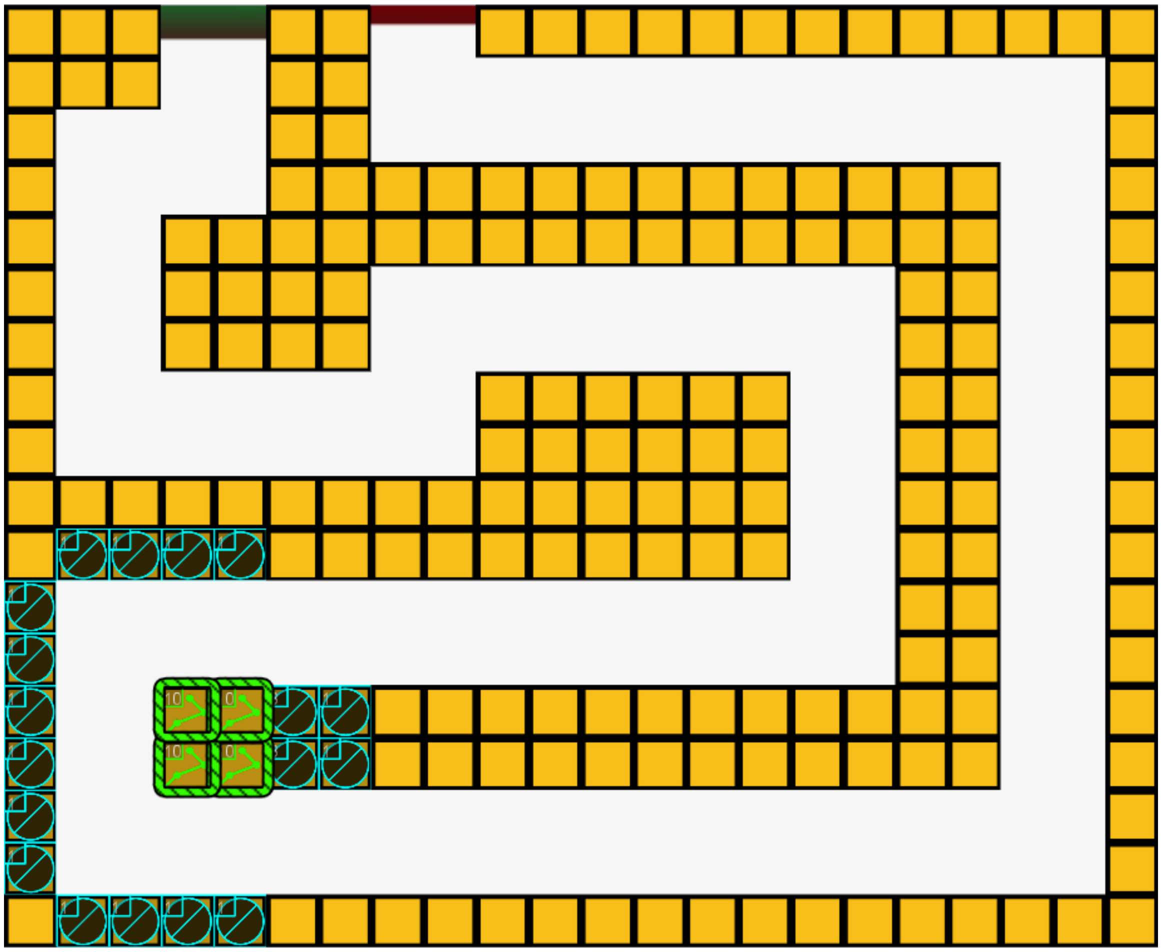


Figure F.7. Perfect Solution to Elementalish. Subject E1100. The subject places their offense towers at the end of a U-turn (SATISFICING STRATEGY: USE U-TURNS). The offense towers are grouped together in an effort to simulate an Area of Effect tower, a type of tower popular in other tower defense games but not present in GopherTD (see STRATEGY E8: CREATE A CORE OF MULTI-TARGET TOWERS). Note that the subject avoids placing slowing towers in the interior corners since towers placed there have less usable range (STRATEGY E1: AVOID SINGLE SIDED INTERIOR CORNERS)

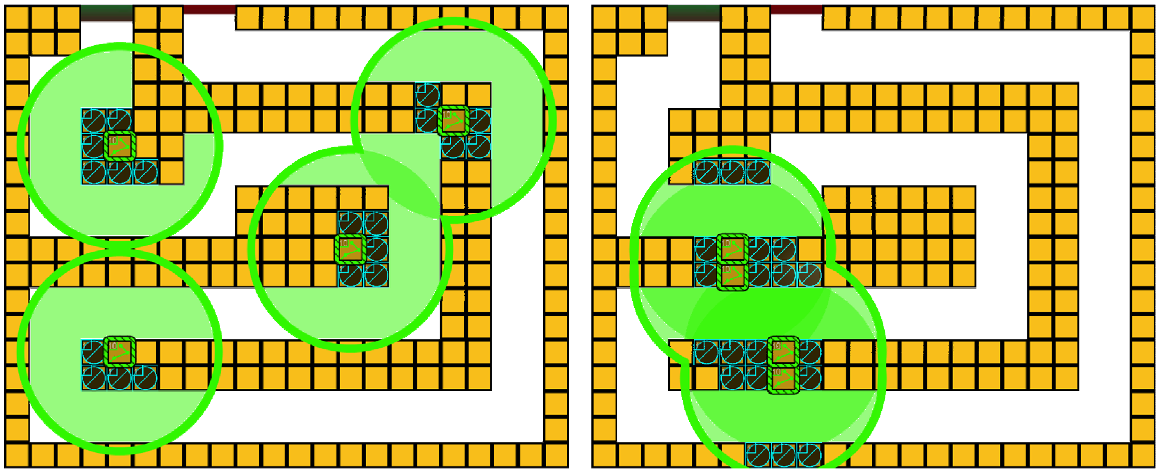


Figure F.8. Perfect Solutions to Elementalish. LEFT: 3N0700 creates four independent kill zones, with offense towers distributed and placed where they cover the most path area. RIGHT: s0400 concentrates their towers near each other, forming one three kill zones – one where all offense towers participate, two where one group of offense towers are active. S0400 sacrifices usable range in order to get attack window separation.

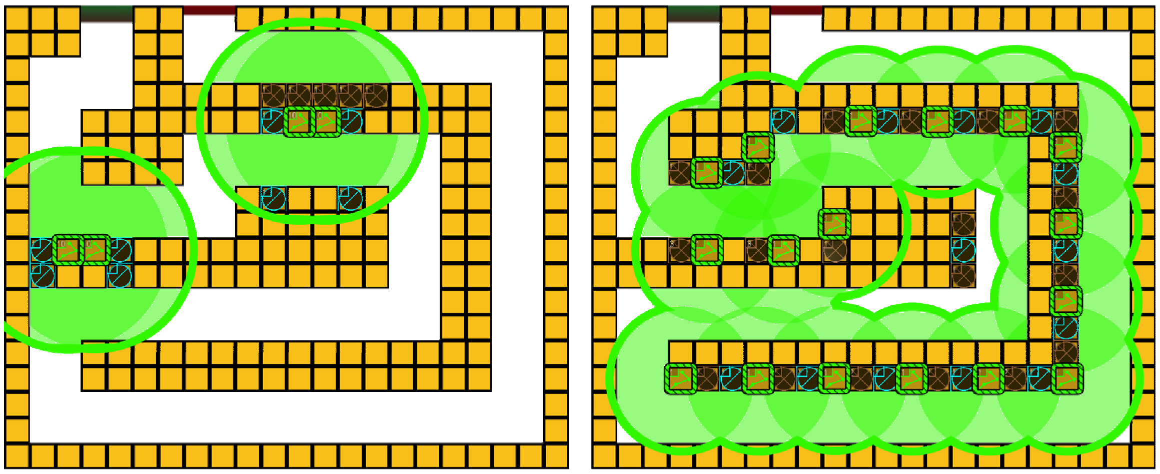


Figure F.9. Perfect Solutions to Elementalish. LEFT: E1900 sacrifices usable range in order to get attack window separation. Five freezing towers are placed by the exit to catch leaked creeps. Two of the primary offense towers are able to be re-used as cleanup towers. RIGHT: e2100 uses 17 low-powered (level 1) fast attack towers spaced out in one long continuous chain, separated by slowing and freezing towers.

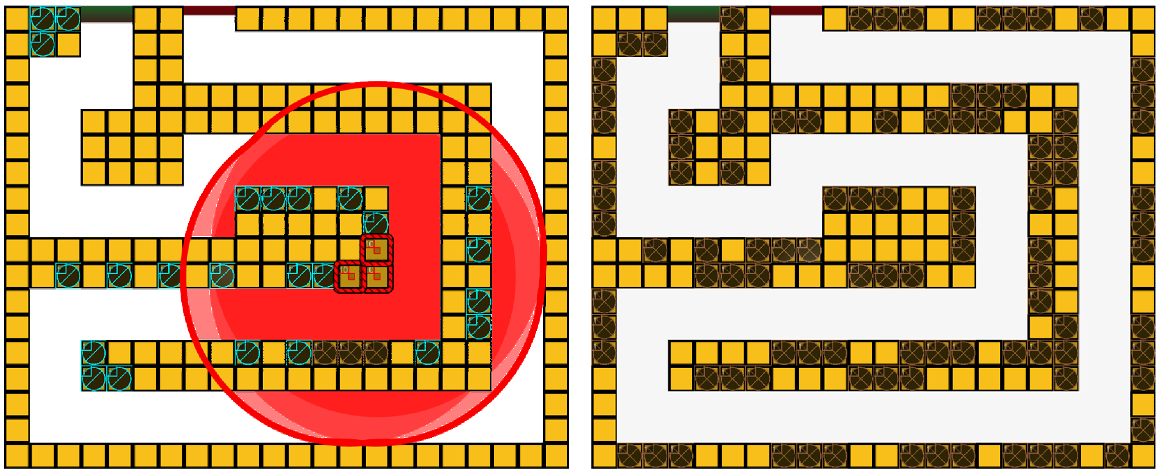


Figure F.10. Perfect Solutions to Elementalish. LEFT: a0200 places three high-powered (level 10) strong attack towers where they can both cover the center of the map and reach two lanes outside the center. Three slowing towers are put at the map entrance to slow down one line of creeps (the one that would be on the outside prior to the kill zone). Three freezing towers are placed near the exit of the first attack window and able to cover the second attack window in order to catch leaked creeps. RIGHT: e2500 was one of two subjects that consistently used a swamp strategy, placing freezing towers in sets of three distributed throughout the map. While freezing towers have very little attack power, if they're able to hold the creeps long enough, the damage will eventually destroy the creeps.

F.2.2 Finger

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	-	26.15	8.19	19.91
Range	-	-	22-28	0-27	14-28
Standard Deviation	-	-	1.90	6.09	4.26
Perfect Scores	-	-	8/20	-	-

Table F.8. Scores for Finger, with Benchmarks.

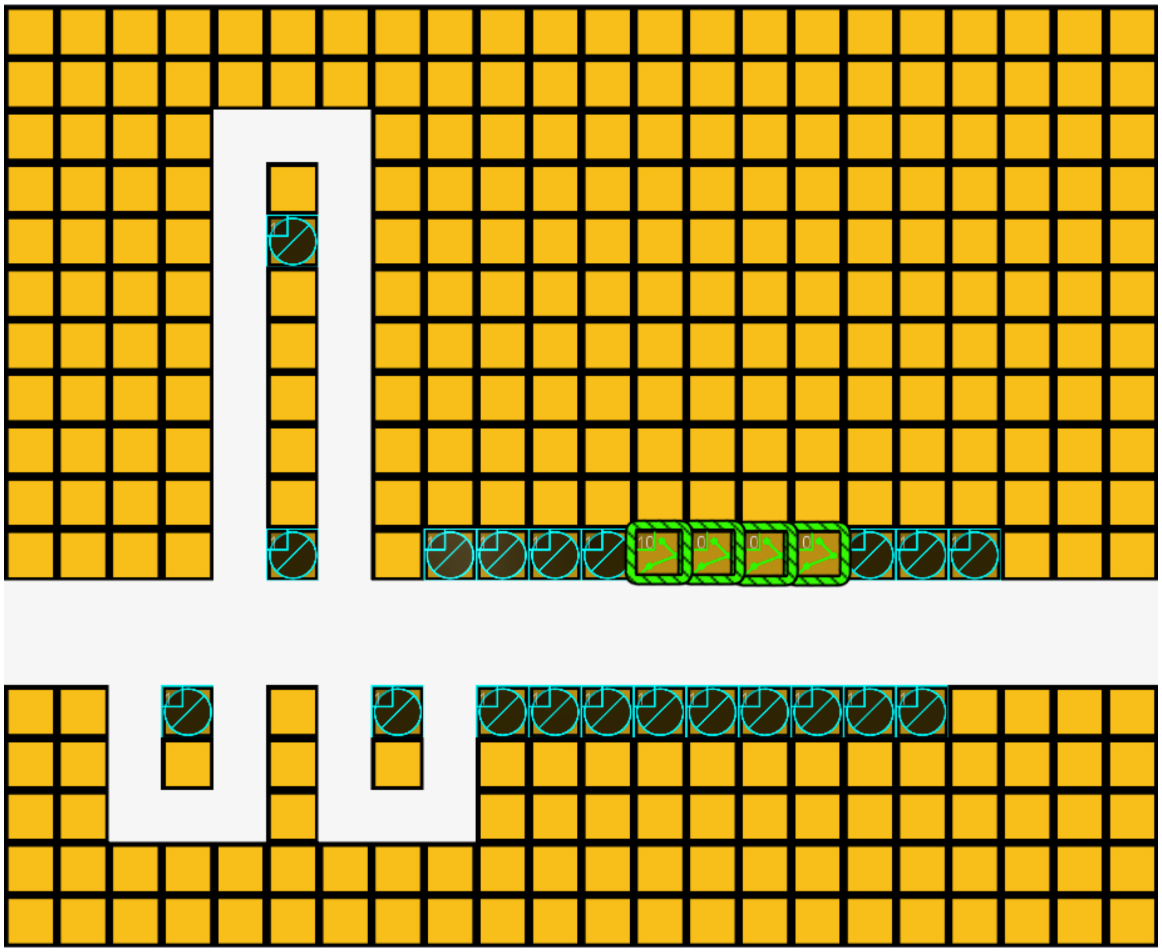


Figure F.11. Perfect Solution to Finger. On Finger, the top row of creeps moves directly to the exit while the bottom row follows a circuitous path through three U-turns. 3E0400 placed all their offense towers where the top and bottom row of creeps would have completely separated (STRATEGY TP8: EXPLOIT GEOMETRY), allowing the towers to concentrate all their attack power on the one line of creeps. Slowing towers are placed near the offense towers to give the offense towers more time to attack (STRATEGY TP1: SLOW IN RANGE). To make sure that the bottom line of creeps doesn't enter the kill zone before the offense towers are done with the top line, slowing towers are placed where they will slow down the bottom line (STRATEGY TP11: TARGET SELECTION-BASED DIFFERENTIAL SLOWING, STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING).

F.2.3 No Support

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	-	25.65	3.16	21.49
Range	-	-	17-28	0-16	8-28
Standard Deviation	-	-	2.54	4.23	5.45
Perfect Scores	-	-	3/20	-	-

Table F.9. Scores for No Support, with Benchmarks.

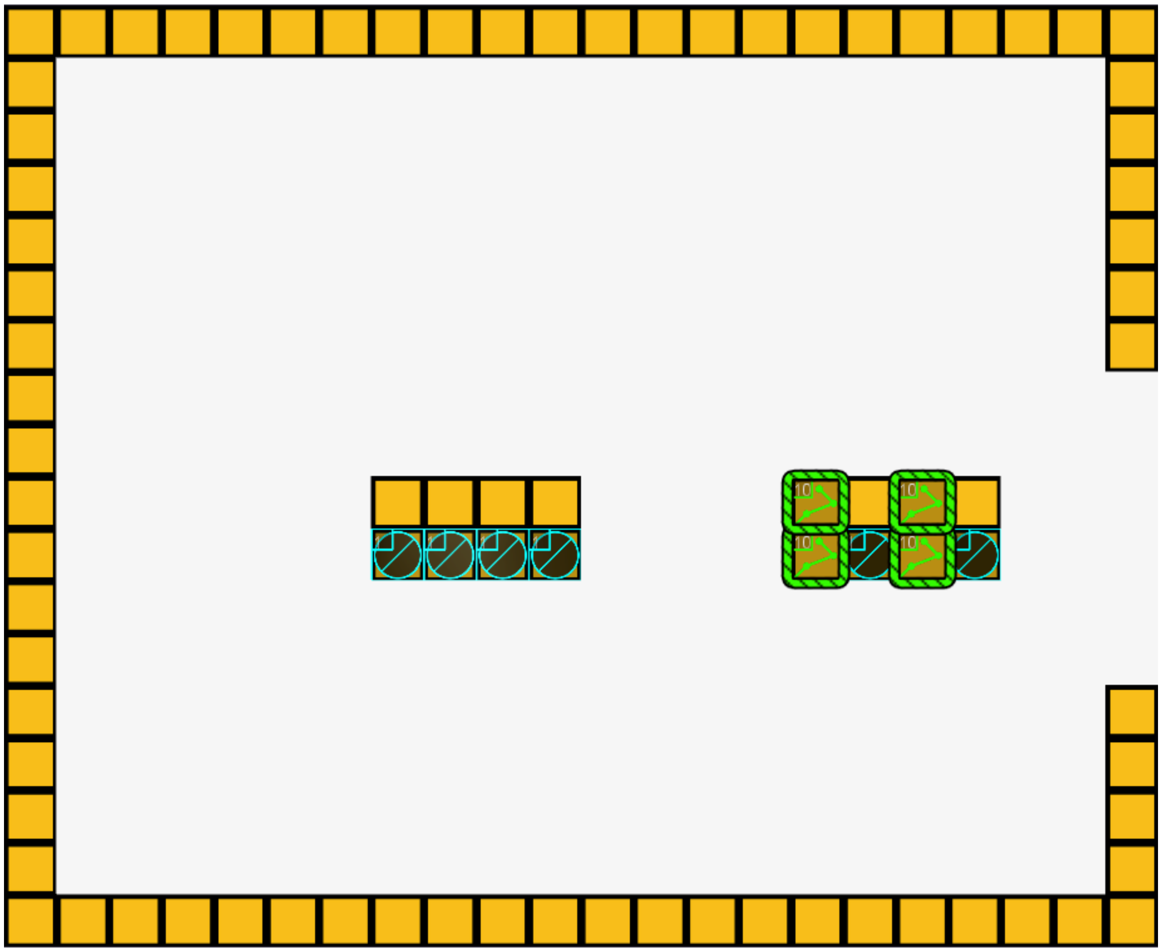


Figure F.12. Perfect Solution to No Support. Subject a0100's solution places the slowing towers so that they only slow the inside line (STRATEGY TP10: PRE-KILL ZONE LIMITED VIEW DIFFERENTIAL SLOWING). The outside line pulls ahead, allowing the offense towers to focus on the inside line. The outside line arrives back in range just as the inside line is leaving, allowing the offense towers to focus on the outside line. To make this possible, slowing towers at the U-turn are used to slow the outside line when they get close to the offense towers (STRATEGY TP6: SLOW BETWEEN ATTACK WINDOWS). The inside line arrives after the outside line is destroyed. The inside line fully exits the kill zone before re-entering it, giving the towers time to "reset" (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION). Note that while subjects in the third experiment were given 20 blue slowing towers, this solution only used six. The 14 towers not used were worth 4,200, 8.5% of their total budget.

F.2.4 Slim Pickings

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	22.84	27.10	20.32	26.01
Range	-	11-28	17-28	3-28	15-28
Standard Deviation	-	5.78	2.49	6.76	2.92
Perfect Scores	-	11/25	15/20	-	-

Table F.10. Scores for Slim Pickings, with Benchmarks.

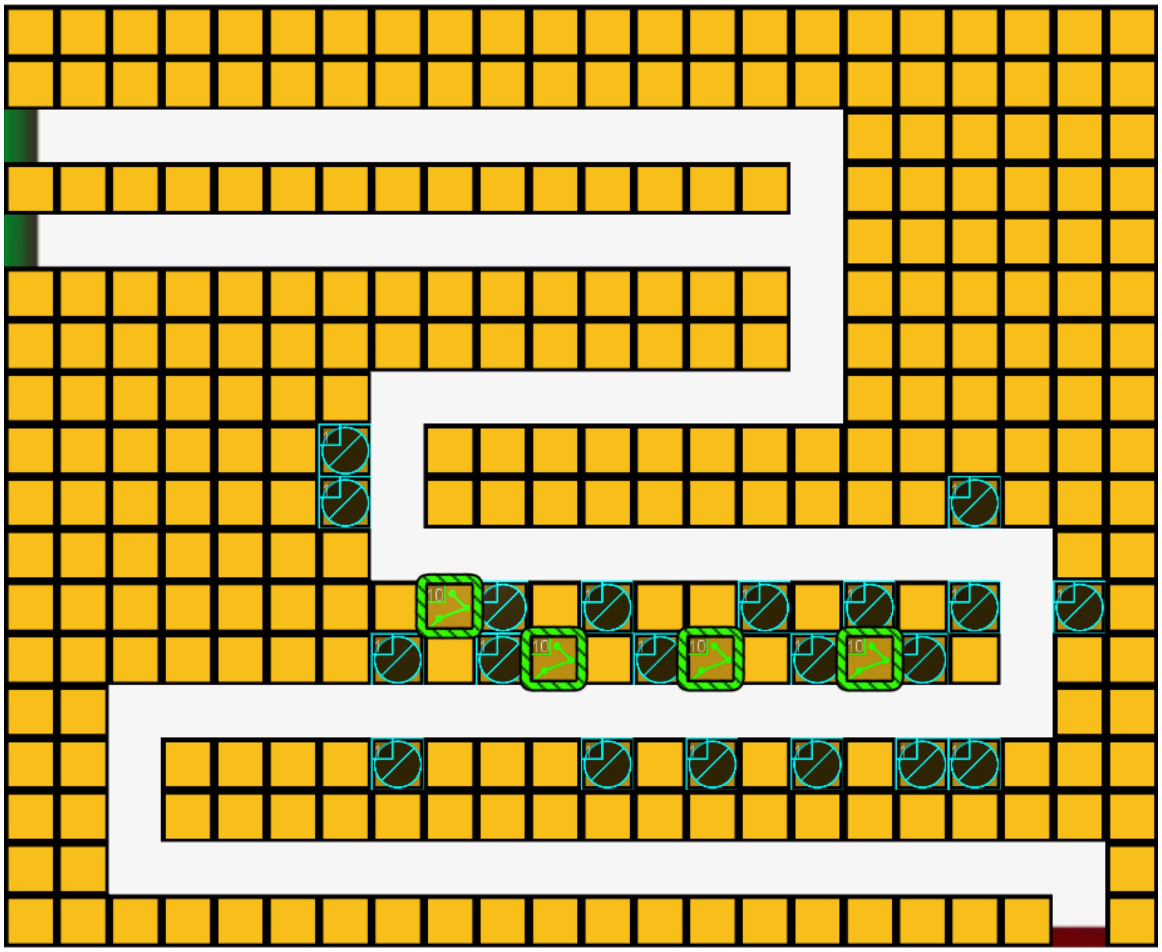


Figure F.13. Solution to Slim Pickings (3N0900). The subject placed their offense towers in a chain along a column in the center of the map, forming a single kill zone. This area is surrounded by slowing towers to slow the creeps while they're in range of the offense towers.

F.2.5 Snaking Path

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	23.54	24.28	27.50	23.49	25.03
Range	7-28	12-28	26-28	10-28	15-28
Standard Deviation	7.36	5.05	0.69	4.15	3.25
Perfect Scores	6/13	11/25	12/20	-	-

Table F.11. Scores for Snaking Path, with Benchmarks.

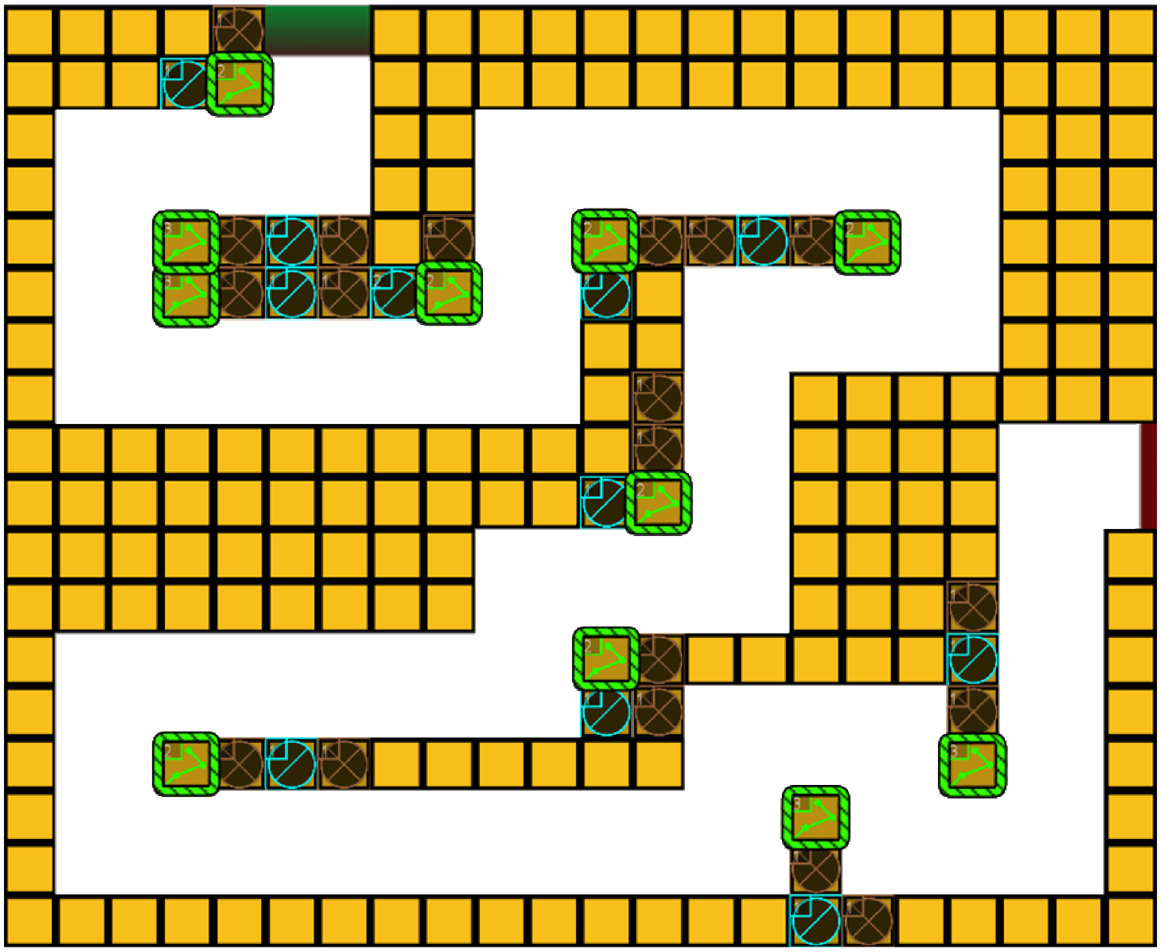


Figure F.14. Perfect Solution to Snaking Path. e2800 placed a series of low-powered (level 2 and 3) fast attack Green3 towers around the map, making sure that the creeps were always being attacked (STRATEGY O7: KEEP ATTACKED AT ALL TIMES). All offense towers are placed on U-turns (SATISFICING STRATEGY: USE U-TURNS) and, when the U-turns are gone, corners (SATISFICING STRATEGY: USE CORNERS) and surrounded by at least one blue slowing and one brown freezing tower (STRATEGY TP1: SLOW IN RANGE).

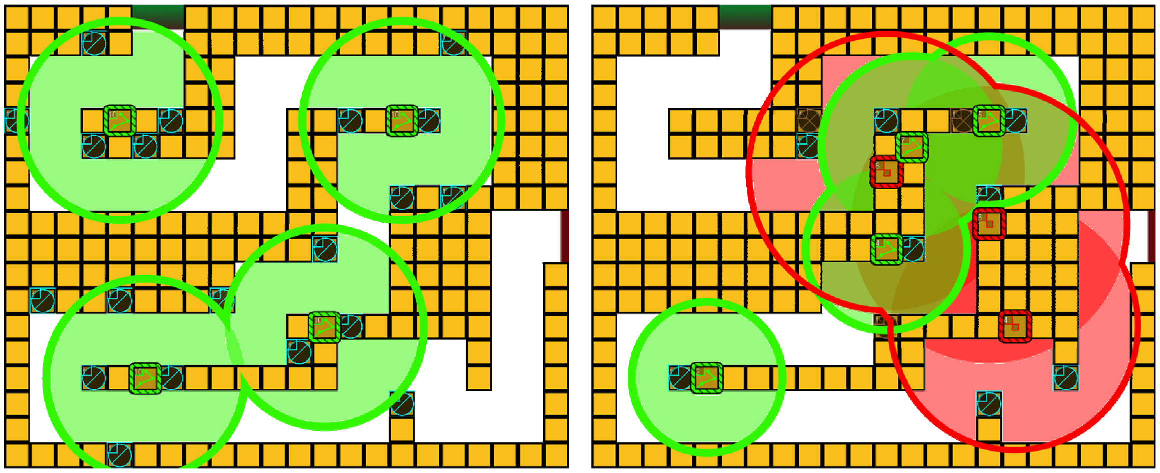


Figure F.15. Perfect Solutions to Snaking Path. LEFT: 3E0200 places their offense towers where they can cover the most path area. RIGHT: e2400 uses a variety of medium-powered fast-but-weak and slow-but-powerful offense towers. Each tower is upgraded to a different level – the subject stopped upgrading towers when it caused the additional range to primarily cover non-usable wall area. A very small number of slowing and freezing towers are used.

F.2.6 The Frog

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	19.88	25.25	16.60	18.57
Range	-	7-28	20-28	9-25	8-26
Standard Deviation	-	5.89	2.63	3.60	3.43
Perfect Scores	-	3/25	5/20	-	-

Table F.12. Scores for The Frog, with Benchmarks.

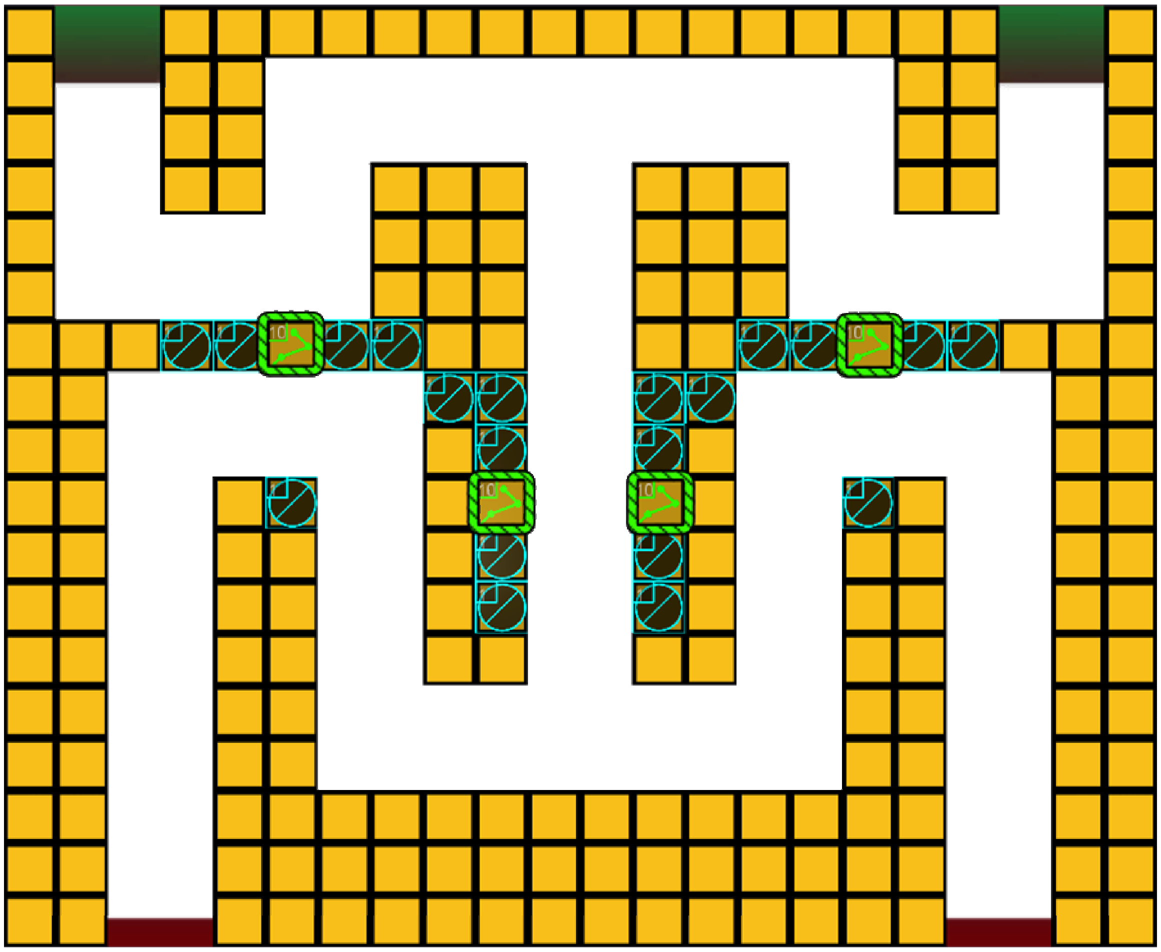


Figure F.16. Perfect Solution to The Frog. 3E0600 places their towers in an unusual chain; the towers are touching but don't follow the path, so creeps aren't always being affected by the chain. Towers are spread out across the chain rather than grouped together (STRATEGY O8: SPREAD OUT OFFENSE TOWERS). Two offense towers having overlapping ranges, able to attack the same creeps. Each of the other two are dedicated to a single line of creeps (STRATEGY E7: DECOMPOSE PROBLEM). Offense towers are placed in the center of walls rather than on U-turns and corners, sacrificing usable range to subdivide and separate their attack windows (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION). Like many subjects, 3E0600 placed towers symmetrically (STRATEGY P4: PLACE TOWERS SYMMETRICALLY), in part because of aesthetics. Slowing towers are placed around the offense towers to give them more time to work (STRATEGY TP1: SLOW IN RANGE).

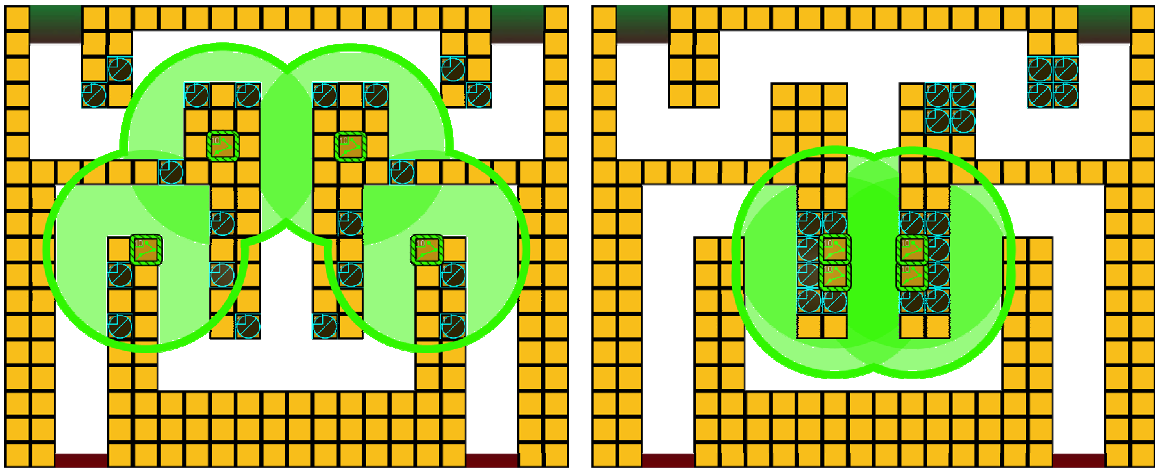


Figure F.17. Perfect Solutions to The Frog. LEFT: 3E0300 spreads their towers out, placing each offense tower where it covers the most path area (STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME). RIGHT: s0400 concentrates all their towers in a single location. To make the offense towers job easier, they slow the right line of creeps, allowing the offense towers to deal with the left line before they have to start on the right line (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING).

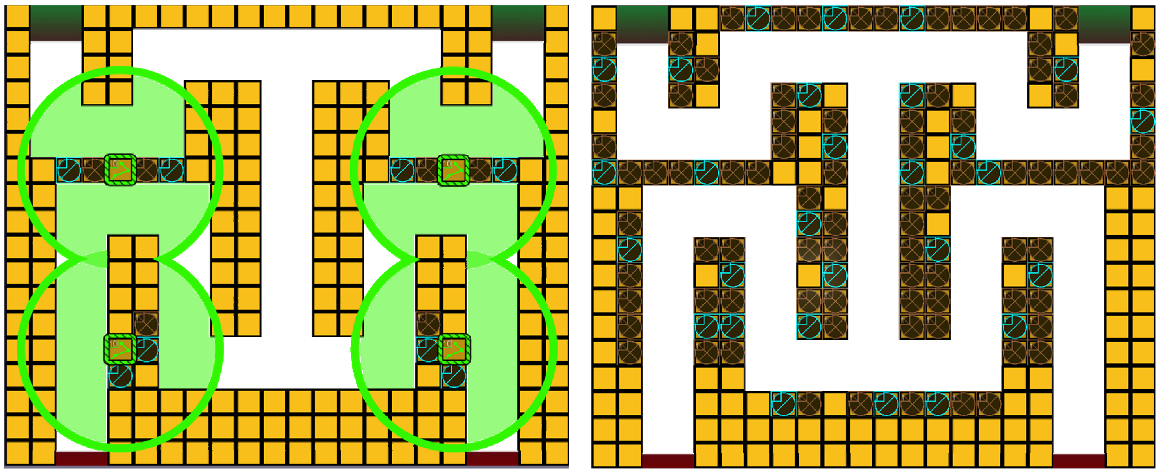


Figure F.18. Perfect Solutions to The Frog. LEFT: E1900 deals with each line separately with no overlap. Offense towers are placed so that the line of creeps must completely exit their range before re-entering it (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION). RIGHT: e2500 uses no offensive towers, relying on a large number of freezing towers to permanently hold the creeps in place, allowing even low power towers to eventually destroy the creeps (STRATEGY TP2: SWAMP).

F.2.7 Up and Down

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	22.00	23.64	26.75	25.78	26.17
Range	8-28	13-28	15-28	18-28	16-28
Standard Deviation	7.76	5.27	2.88	2.69	2.47
Perfect Scores	6/13	11/25	11/20	-	-

Table F.13. Scores for Up and Down, with Benchmarks.

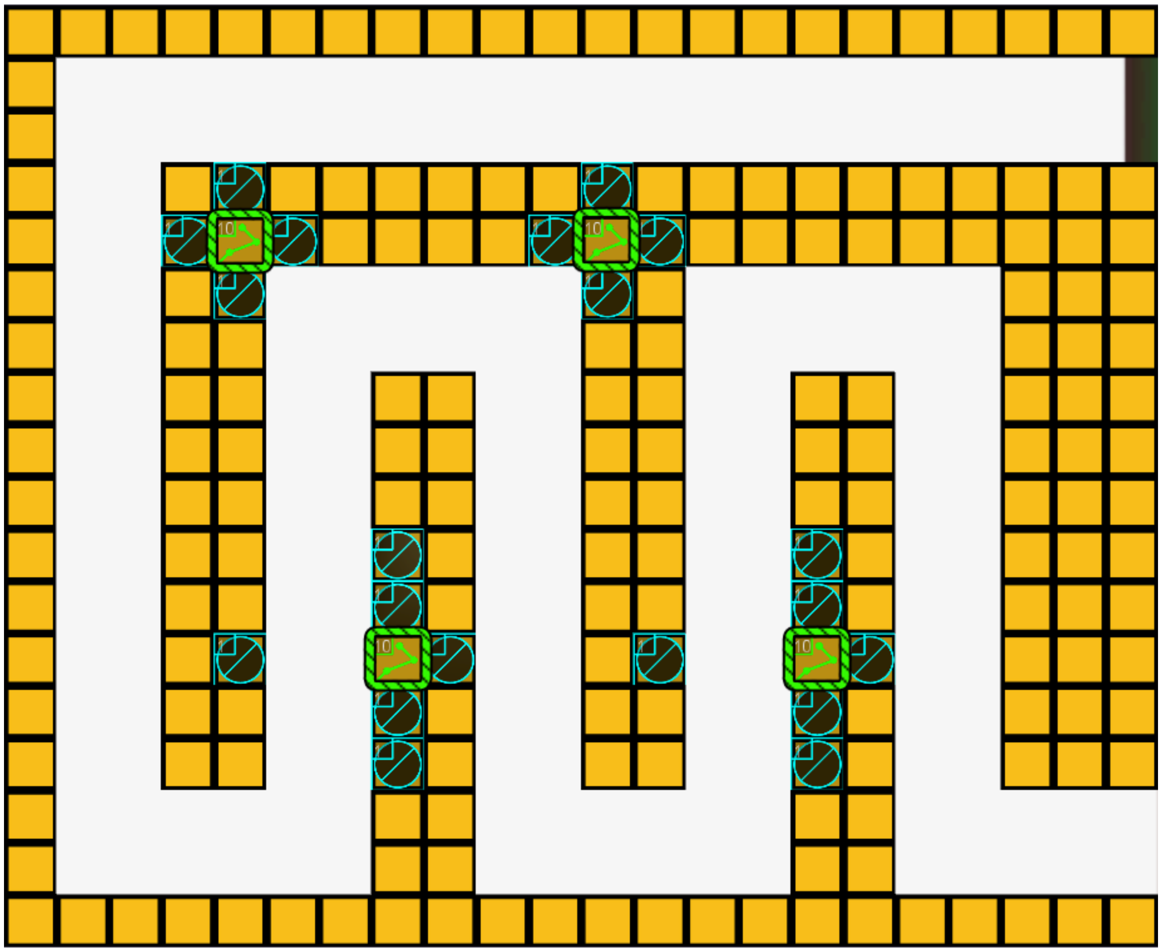


Figure F.19. Perfect Solution to Up and Down. 3E0600 creates four cross-shaped kill zones. Offense towers are placed in the center of walls rather than on U-turns and corners, sacrificing usable range to subdivide and separate their attack windows (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION). Like many subjects, 3E0600 placed towers symmetrically (STRATEGY P4: PLACE TOWERS SYMMETRICALLY), in part because of aesthetics. Slowing towers are placed around the offense towers to give them more time to work (STRATEGY TP1: SLOW IN RANGE).

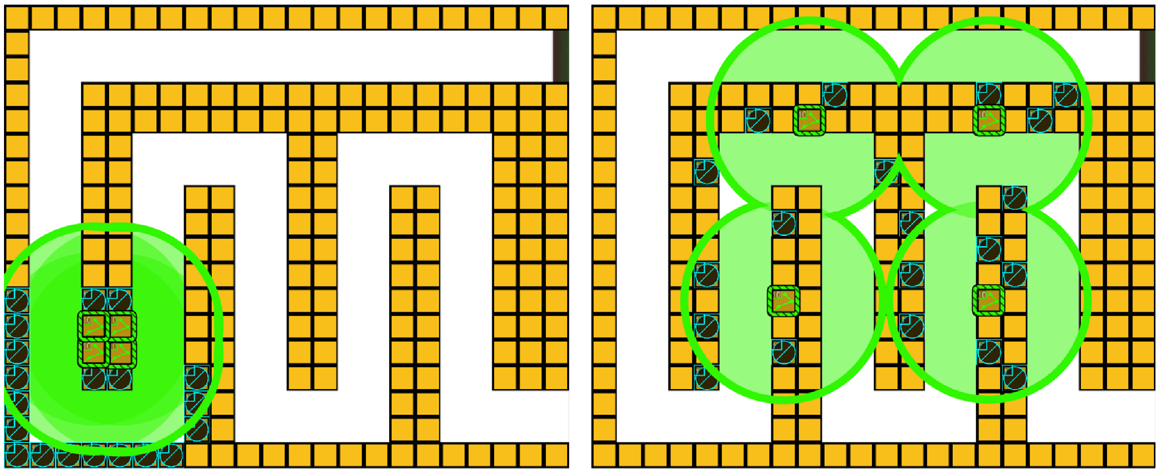


Figure F.20. Perfect Solutions to Up and Down. LEFT: E0100 concentrates all of their towers into a single U-turn-based kill zone. RIGHT: 3E0100 spreads their offense towers out, placing them in the center of walls in order to separate their attack windows.

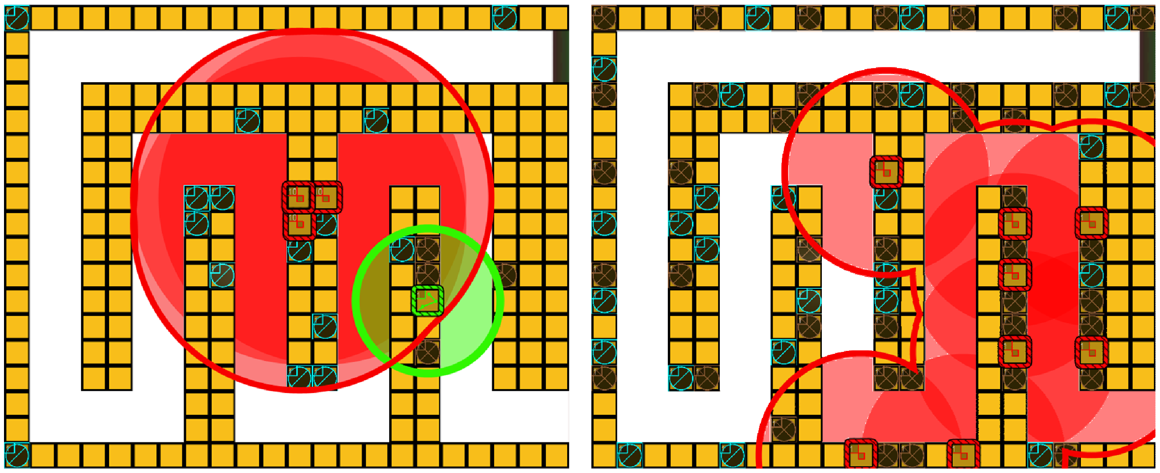


Figure F.21. Perfect Solutions to Up and Down. LEFT: a0100 places three slow-but-powerful red towers where they can cover the majority of the map. A low-powered (level 1) fast attack tower and several freezing towers form a cleanup area at the exit of the kill zone's exit. Three slowing towers are placed on the map corners to slow the (initially) outside line of creeps. RIGHT: E1700 uses a series of low-level slow-but-powerful red towers placed along the last third of the path.

F.2.8 Void

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	14.23	19.96	22.80	14.25	18.63
Range	3-28	9-28	18-28	3-24	9-28
Standard Deviation	9.15	5.33	3.25	4.84	4.47
Perfect Scores	2/13	2/25	3/20	-	-

Table F.14. Scores for Void, with Benchmarks.

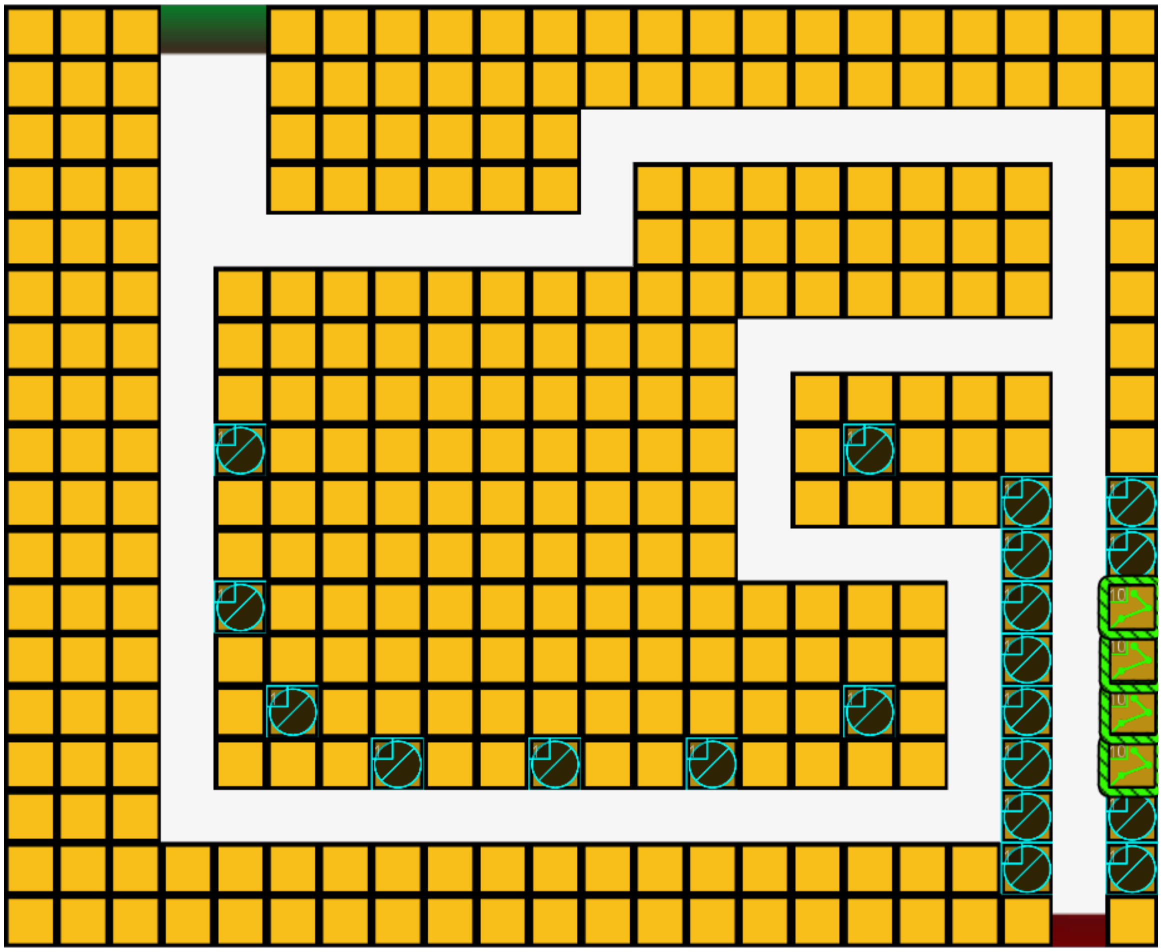


Figure F.22. Perfect Solution to Void. s0400 places a concentrated row of fast attack towers by the map exit. Because one line of creeps follows a significantly longer path than the other, the offense towers will only deal with one line at a time (STRATEGY TP8: EXPLOIT GEOMETRY). To make sure the second line doesn't arrive while the kill zone is busy with the first line, slowing towers are placed along its path (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). The offense towers are placed on the right side of the exit rather than the left. On the left, the towers would cover twice as much path area and get two attack windows but would be equally close to each line of creeps, causing the towers to thrash between the creeps about to exit the map and creeps that can be attack later. The subject sacrifices the range and attack windows in order to maintain focus on the creeps closest to exiting the map (STRATEGY T7: SACRIFICE ATTACK WINDOW).

F.3 Hard Maps

	2 in 1 out			Bottleneck		
	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3
Average Score	12.15	14.76	17.95	12.46	15.20	20.35
Range	5-24	7-24	14-23	2-26	6-26	17-26
Standard Deviation	5.52	4.46	2.54	6.53	5.12	2.06

	Do the Splits			Ladder		
	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3
Average Score	-	-	13.90	-	18.44	23.70
Range	-	-	8-21	-	9-27	20-27
Standard Deviation	-	-	3.48	-	5.58	2.08

	No Left Turns			Pathways		
	Exp. 1	Exp. 2	Exp. 3	Exp. 1	Exp. 2	Exp. 3
Average Score	-	-	15.55	-	15.60	19.10
Range	-	-	11-28	-	6-28	10-28
Standard Deviation	-	-	4.50	-	5.99	5.92

Table F.15. Scores for Difficult Maps.

Only two of the six hard maps were solved during the experiments. In this section we both show solutions and trace through their execution to show how they behave. Solutions for the four unsolved maps were created the author. All required six attempts to find a working solution. A good score was obtained on the first attempt and the following five attempts were used to optimize small details that eventually led to perfect solutions. That it took exactly six tries to earn a perfect score on each map is purely coincidence.

F.3.1 2 in 1 out

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	12.15	14.76	17.95	3.44	17.21
Range	5-24	7-24	14-23	0-18	8-24
Standard Deviation	5.52	4.46	2.54	4.35	2.91
Perfect Scores	0/13	0/25	0/20	-	-

Table F.16. Scores for 2 in 1 out, with Benchmarks.

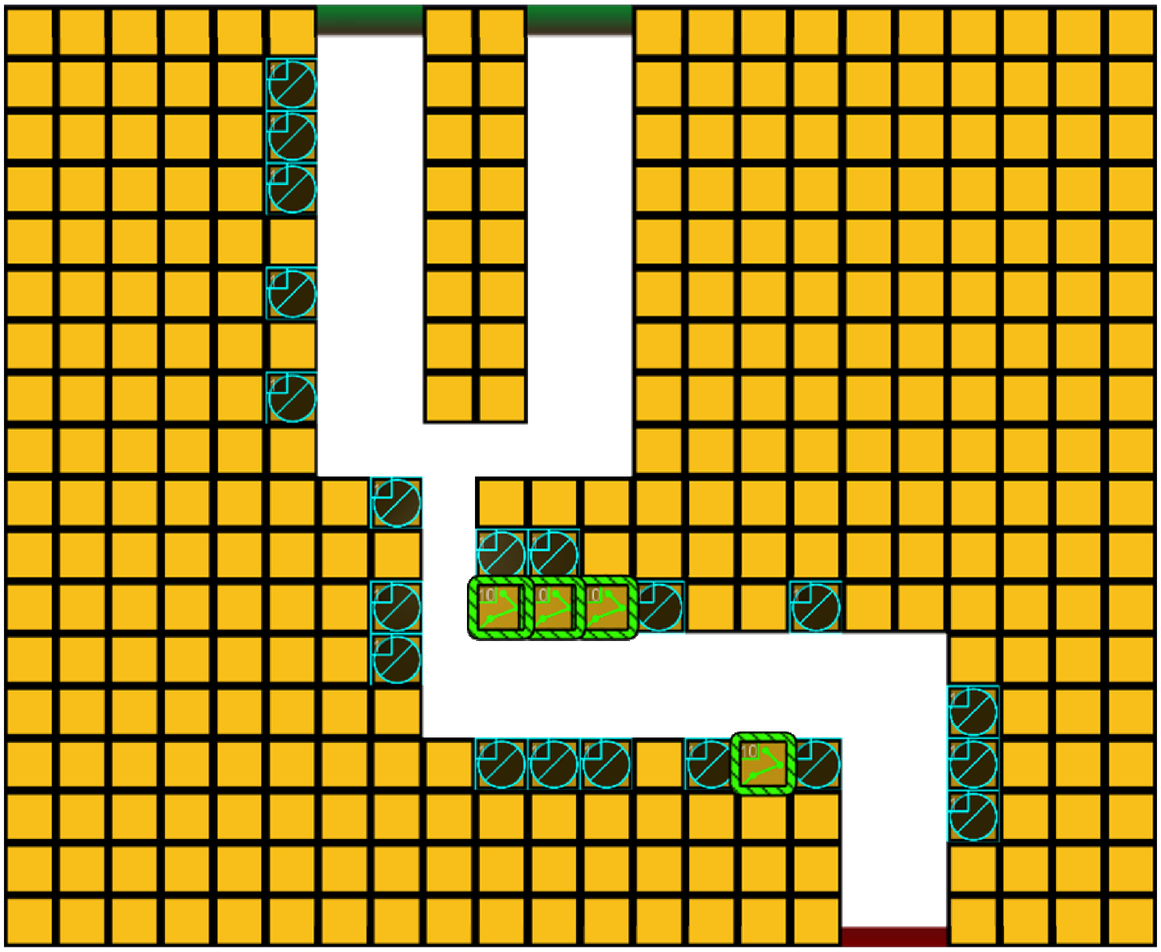


Figure F.23. Perfect Solution to 2 in 1 Out. Created by author. Fast towers are used because the paths are too short to give the slow-but-strong red towers enough time to fire 28 times, even if the creeps were slowed. Blue slowing towers are placed along the left path to separate them from the creeps on the right path (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). The left path is longer and would normally be the better one to slow but it enters the kill zone first. The three green offense towers in center were moved down one tile from the optimal position for usable range in order to not target creeps in the DIFFERENTIAL SLOWING area. One tower is placed near the exit to catch creeps that make it through the kill zone (STRATEGY P1: USE CLEANUP TOWERS).

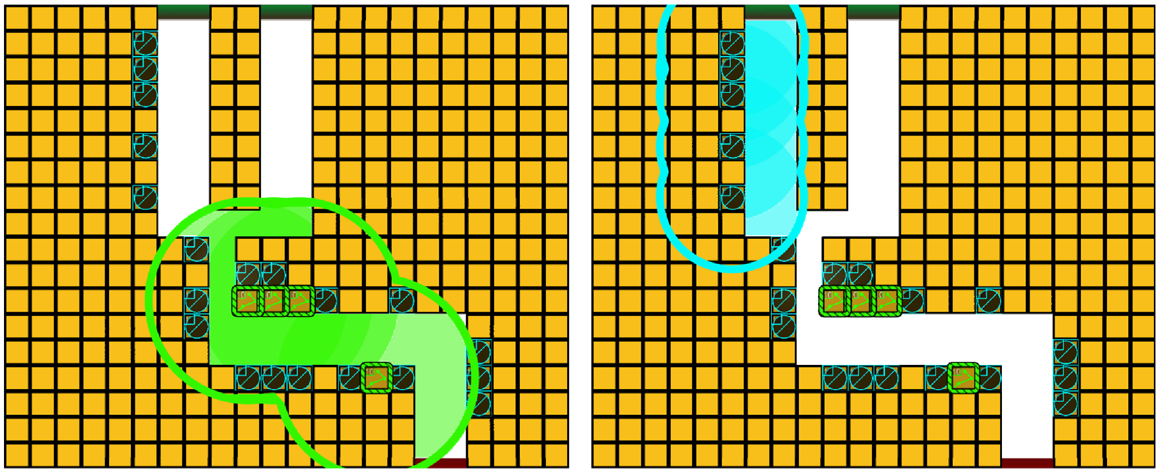
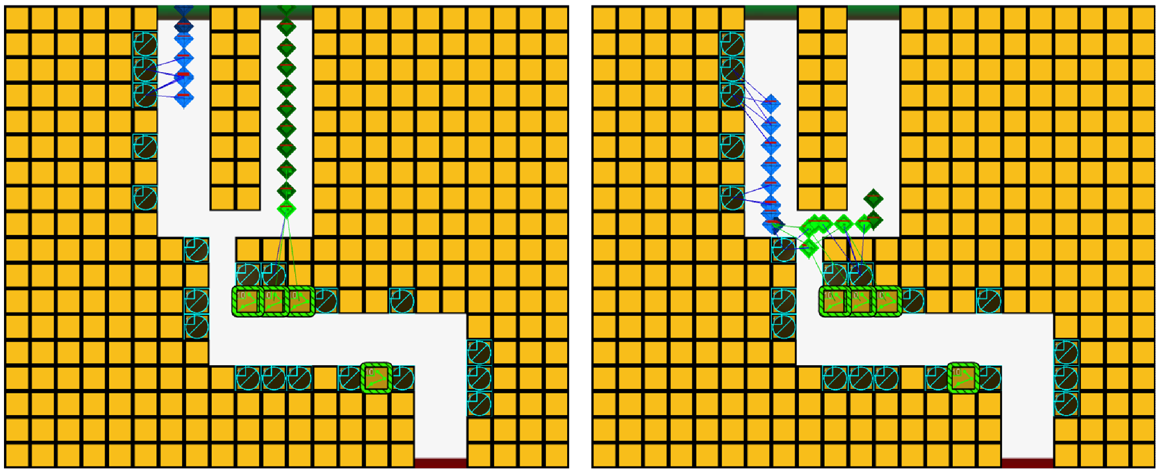
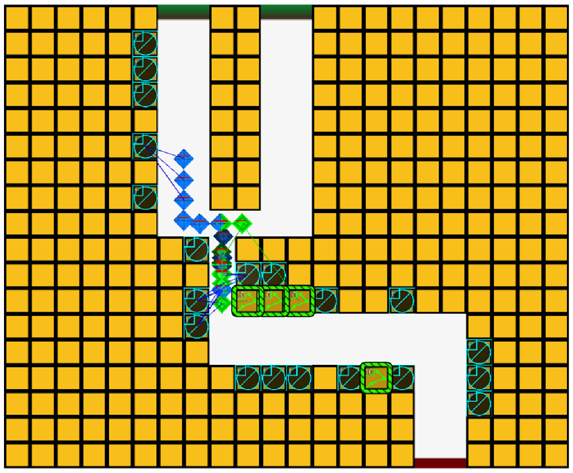
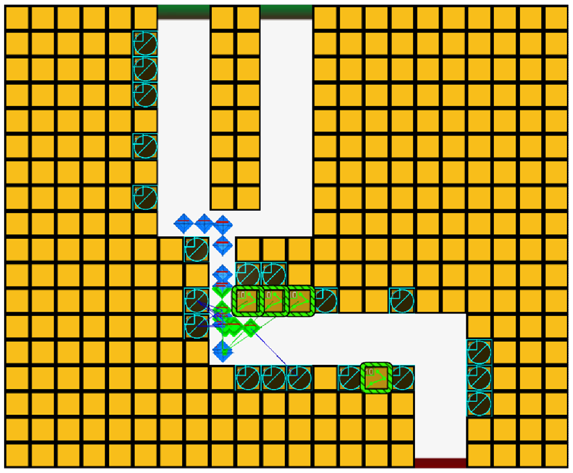


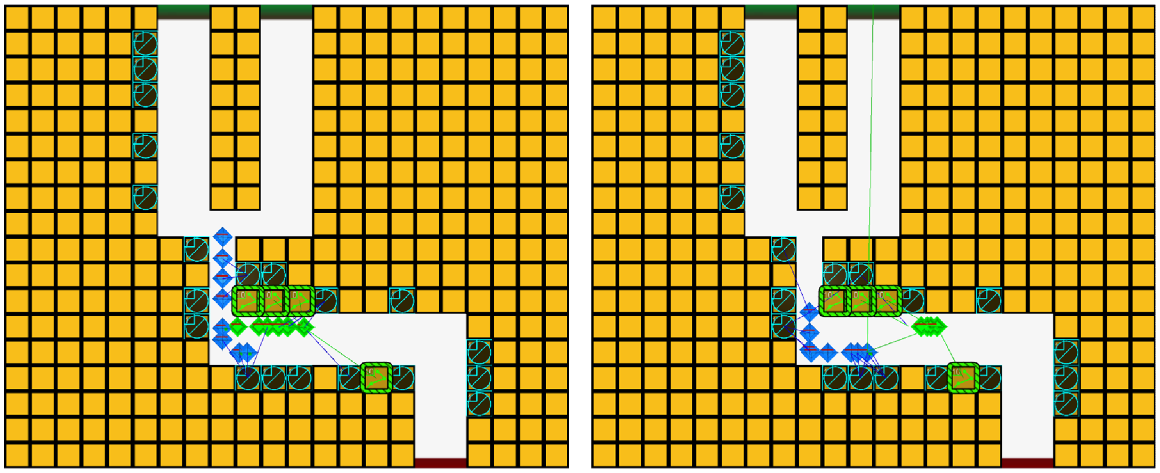
Figure F.24. Execution of Differential Slowing in 2 in 1 Out. LEFT: The area covered by the offense towers. It is lowered so that it does not overlap the DIFFERENTIAL SLOWING area. RIGHT: The area covered by the DIFFERENTIAL SLOWING towers. DIFFERENTIAL SLOWING requires a long path to fully separate so this only slightly separates the two lines of creeps. However, there aren't many better positions to place the five slowing towers as the rest of the path is already covered.

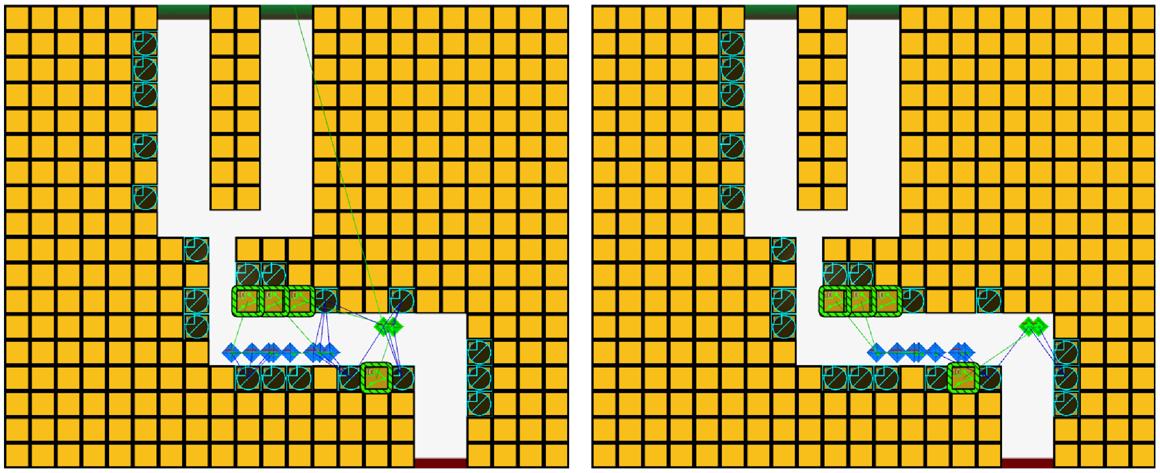


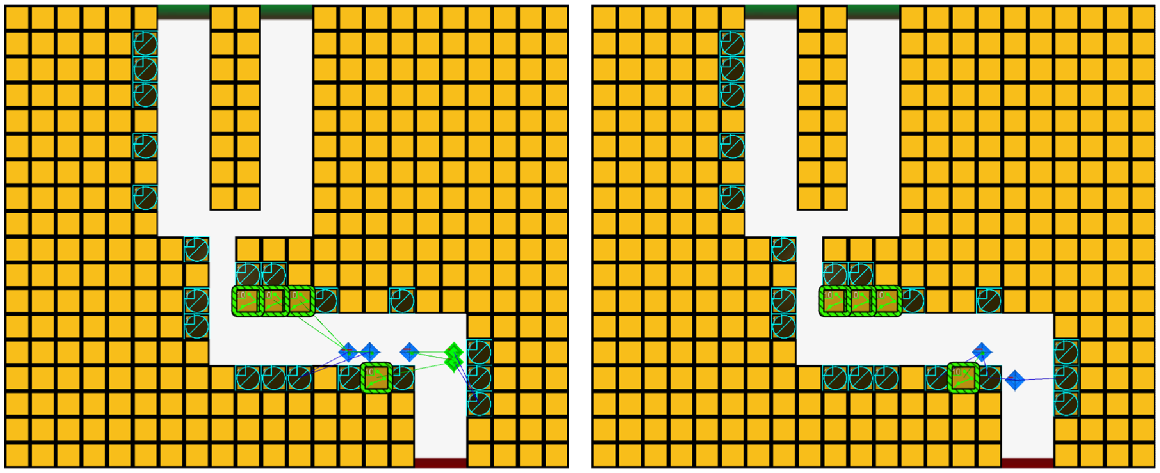
The two lines of creeps enter the map. The left/blue line is slowed, the right/green line is not but takes a path that is one tile longer. The result is that the left/blue line is four creep lengths behind the right/green line.

In the left passage, the creeps overlap, making it hard to tell which creeps the offense towers will target. Coming out of that passage, the left/blue line of creeps turn an outside corner, making their path one tile longer. They are also slowed as soon as they come around the corner. Because all of the offense towers are above the horizontal path area, they will target the right/top/green line first if there are still creeps in that line. To mitigate this, as many green creeps as possible must be stopped before the horizontal area. This is partially done by having the offense towers target the right/green line first when they first enter, partially by delaying the left/blue line to give the offense towers more time with the green creeps and partially by placing slowing towers at the bottom of the horizontal area so that are closer to left/bottom/blue line and therefore slow them first.


LEFT: The top/green line of creeps is slowed by the bottom slowing towers. They would normally target the bottom/blue line but, because of DIFFERENTIAL SLOWING, those creeps were not yet in range. RIGHT: By the time the top/green line is exiting the kill zone, there are four green creeps left and eight blue creeps. The green creeps move into range of the cleanup tower while the kill zone focuses on the bottom/blue line of creeps.


The kill zone has enough time to severely wound most of the bottom/blue line of creeps but not destroy them. Slowing towers keep them slow while the cleanup tower finishes them off. The remaining two green creeps move around an outside corner, making their path two tiles longer than the bottom/blue line.


The cleanup tower finishes off the remaining creeps.

F.3.2 Bottleneck

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	12.46	15.20	20.35	3.86	18.33
Range	2-26	6-26	17-26	0-16	10-26
Standard Deviation	6.53	5.12	2.06	4.22	3.42
Perfect Scores	0/13	0/25	0/20	-	-

Table F.17. Scores for Bottleneck, with Benchmarks.

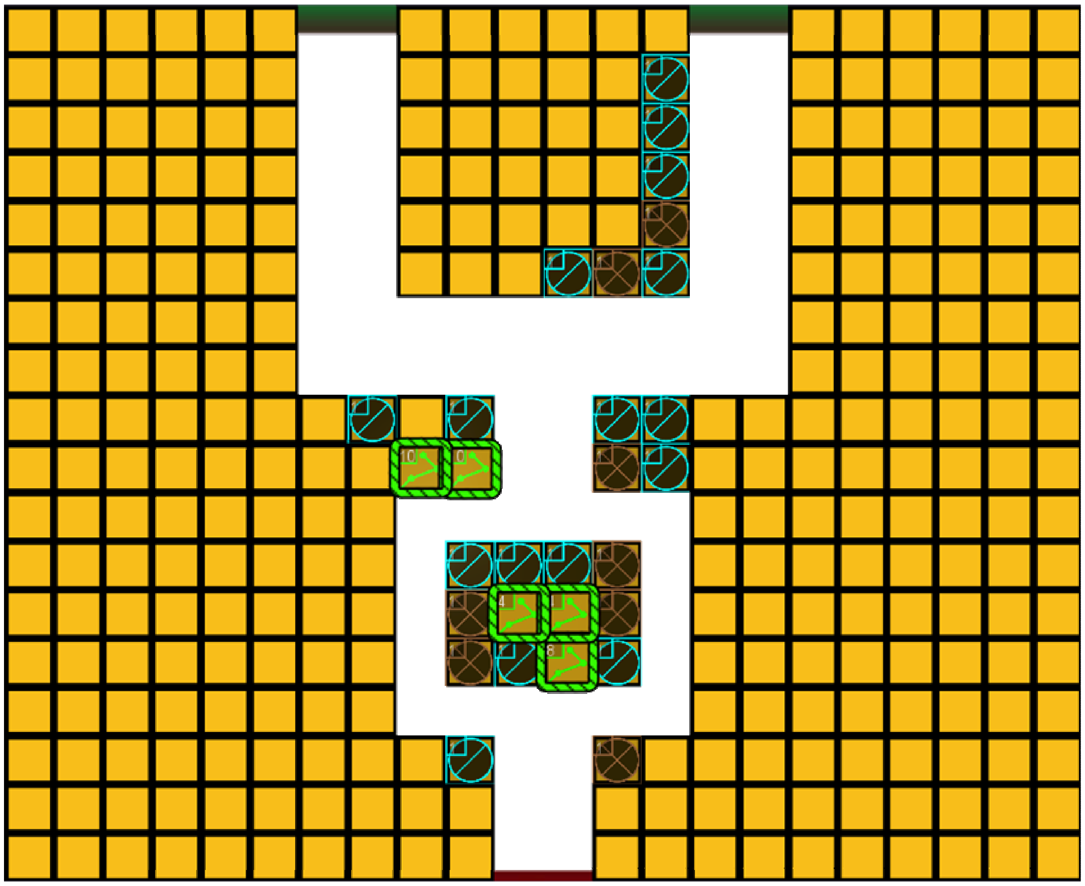


Figure F.25. Perfect Solution to Bottleneck. Created by author. Fast towers are used because the paths are too short to give the slow-but-strong red towers enough time to fire 28 times, even if the creeps were slowed. Blue slowing towers are placed along the right path to separate them from the creeps on the left path (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). Choice of side is arbitrary. Two level 10 Green3 fast attack towers are positioned at the bottleneck. They are on the lower part of the side opposite of the DIFFERENTIAL SLOWING area to keep them from attacking the differentially slowed creeps prematurely. This position allows them to focus on a single line of creeps, which is hard to do in the center area (STRATEGY T9: AVOID ISLANDS). Three green fast attack towers, levels 4, 4 and 8, are in the center area, with the level 8 and one level 4 tower on the right side and the remaining level 4 on the left side. Because the two offense towers at the bottleneck will spend more time with the left line of creeps, less attack power is needed on the left side of the following area than the right. Brown freezing towers in the DIFFERENTIAL SLOWING area are used to hold back a few creeps, decreasing the density of the right line (STRATEGY TP7: SLOW SUBSET TO DECREASE DENSITY). Brown freezing towers are used in the center area to hold a few creeps back from the exit. A brown freezing tower is placed by the exit as a last ditch attempt to stop fleeing creeps (STRATEGY P2: USE FREEZING FOR CLEANUP). The level 8 freezing tower is placed at the bottom of the center area to focus it on the creeps about to exit and not those that were held back by the freezing towers above it (STRATEGY T8: BIAS TO EXIT).

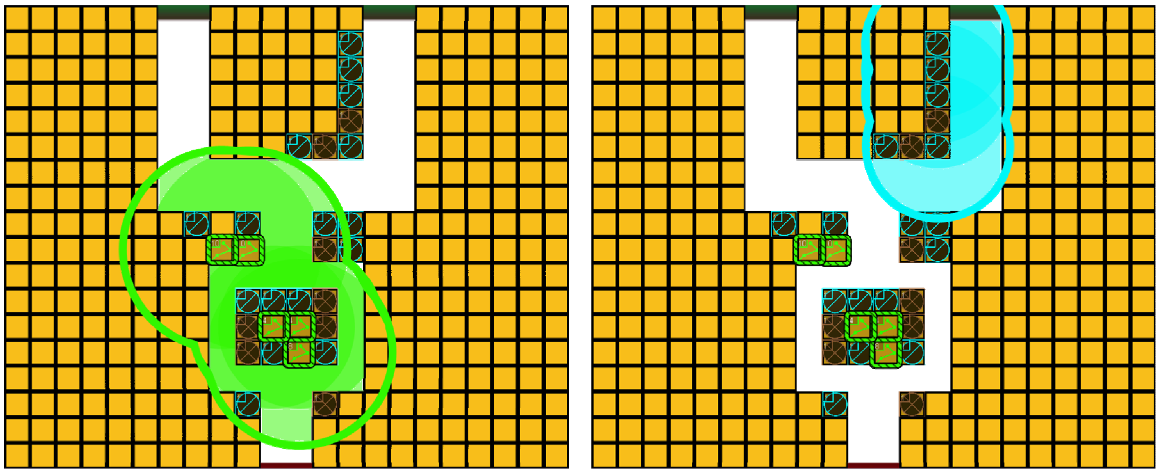
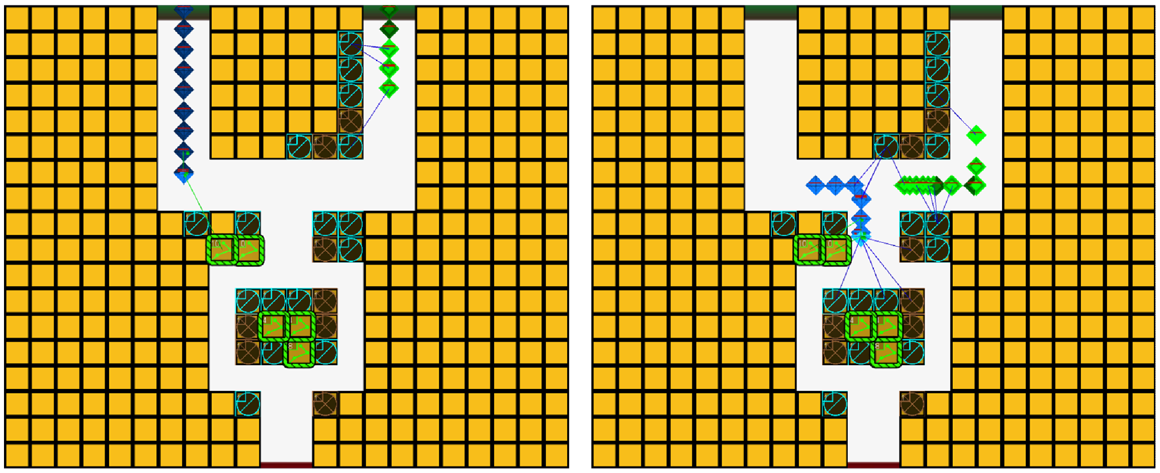
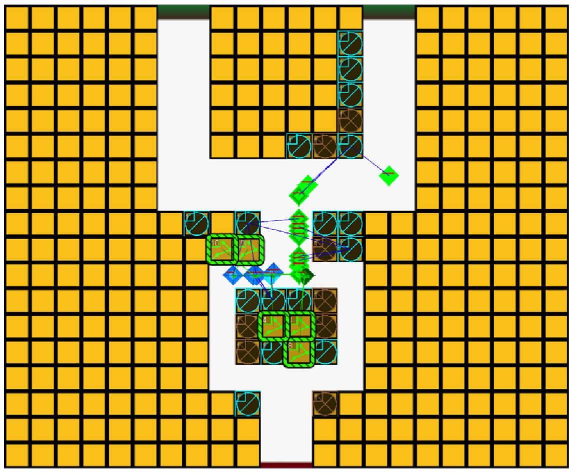
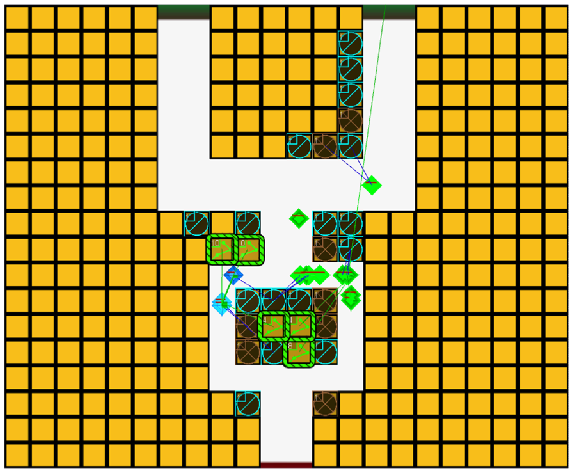


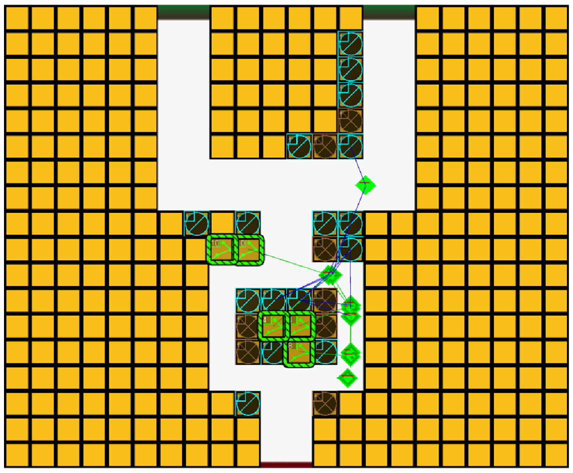
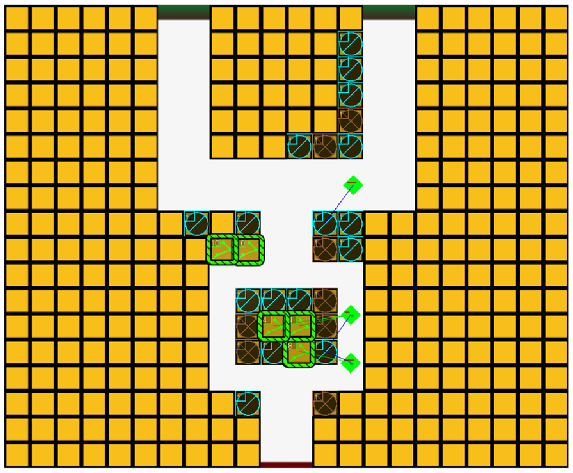
Figure F.26. Execution of Differential Slowing in Bottleneck. LEFT: The area covered by the offense towers. Top offense towers are placed low and to the left to avoid overlapping the DIFFERENTIAL SLOWING area. RIGHT: The area covered by the DIFFERENTIAL SLOWING towers.



The slowing towers on the right side slow the right/green line of creeps. The left/blue line of creeps if four tiles ahead of the right line. The two offense towers at the bottleneck attack the left/blue line, destroying half (7) of the creeps in that line before the right/green line is in range.

The five remaining creeps in the blue/left line move into the center of the map where they are attacked by three offense towers (levels 10, 10, 4). The first of the right/blue line creeps enter the map center where they are attacked by two offense towers (levels 4, 8).

LEFT: The left/blue line of creeps is finished. The two level 10 Green3 fast attack towers attack the latter half of the right/green line. All three offense towers in the center of the map attack the first half of the right/green line. RIGHT: Brown freezing towers hold the last creeps in place. They can only stop one creep, making them less effective than blue slowing towers when there are many creeps present but when there are only a few creeps, the brown freezing towers are able to prevent the creeps from advancing.

F.3.3 Do the Splits

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	-	13.90	8.18	13.65
Range	-	-	8-21	0-16	8-21
Standard Deviation	-	-	3.48	3.53	2.24
Perfect Scores	-	-	0/20	-	-

Table F.18. Scores for Do the Splits, with Benchmarks.

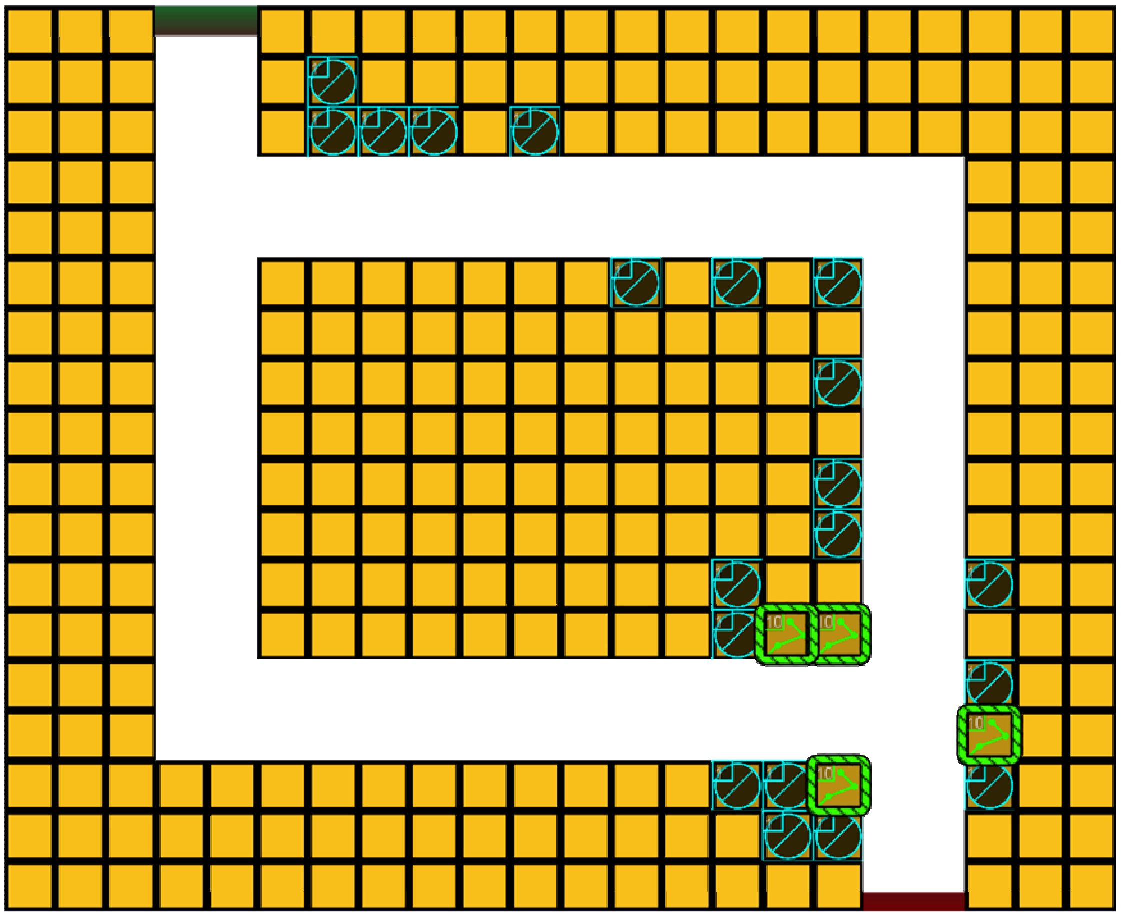


Figure F.27. Perfect Solution to Do the Splits. Created by author. Slowing towers are placed where they only slow the top line of creeps, giving the offense towers more time to focus on the bottom line (STRATEGY TP11: TARGET SELECTION-BASED DIFFERENTIAL SLOWING, STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). Offense towers are placed late in the map to exploit the path synchronization gap created by DIFFERENTIAL SLOWING. Slowing towers are used to slow the creeps while they are inside the kill zone to give the offense towers more time to attack (STRATEGY TP1: SLOW IN RANGE). The first of these slowing towers are precisely placed so that their entrance matches the offense tower's range, neither slowing the creeps too early (STRATEGY E9: DON'T SLOW BEFORE KILL ZONE) nor too late (STRATEGY E10: SLOW AT ENTRANCE TO KILL ZONE). Two offense towers are placed on corners to maximize the area covered (STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME). An offense tower is placed to the left of the corner tower to attack the left/green line. A similar tower is not placed above the corner tower to handle the right/blue line. This is partially because the remaining tower is needed more in a different area, partially because the right/blue line will be slightly weaker because it has been attacked by the DIFFERENTIAL SLOWING towers. The last offense tower is placed on the right wall near the exit. In this location, it loses a significant amount of range but is able to focus on creeps that are about to exit the map without being distracted by less important creeps (STRATEGY T8: BIAS TO EXIT).

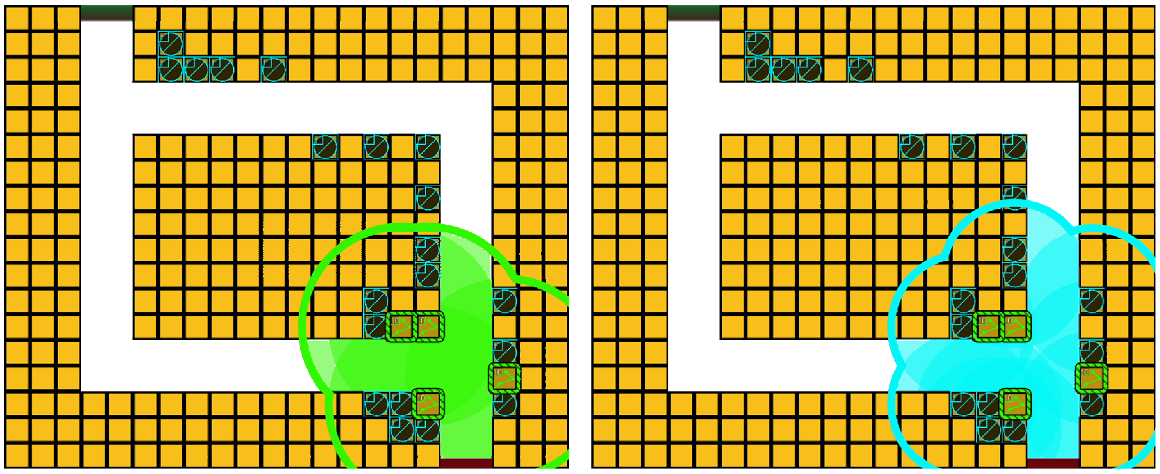


Figure F.28. Kill Zone in Do the Splits. LEFT: The area covered by the offense towers. The left/bottom side's coverage is one tile longer than the right/top side. RIGHT: The area covered by the SLOW IN RANGE towers.

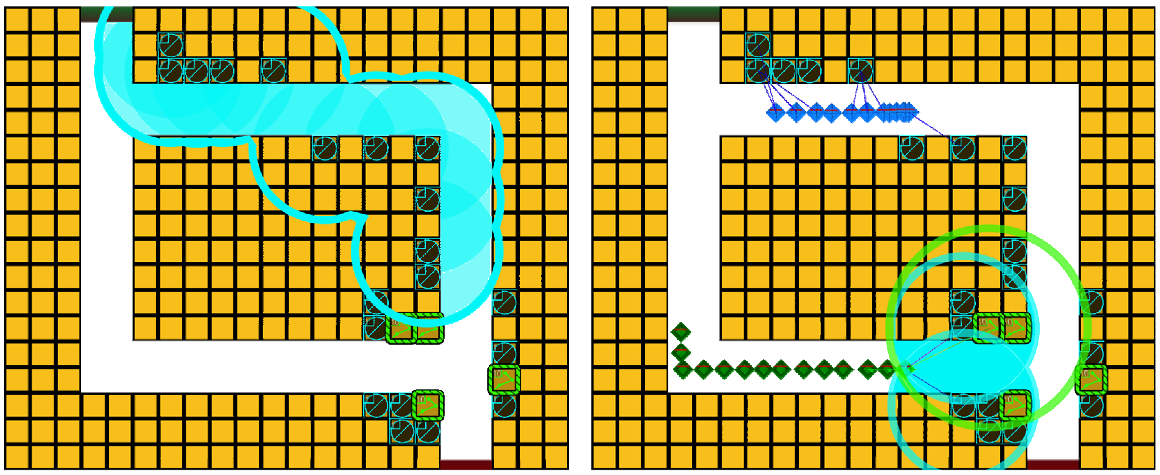
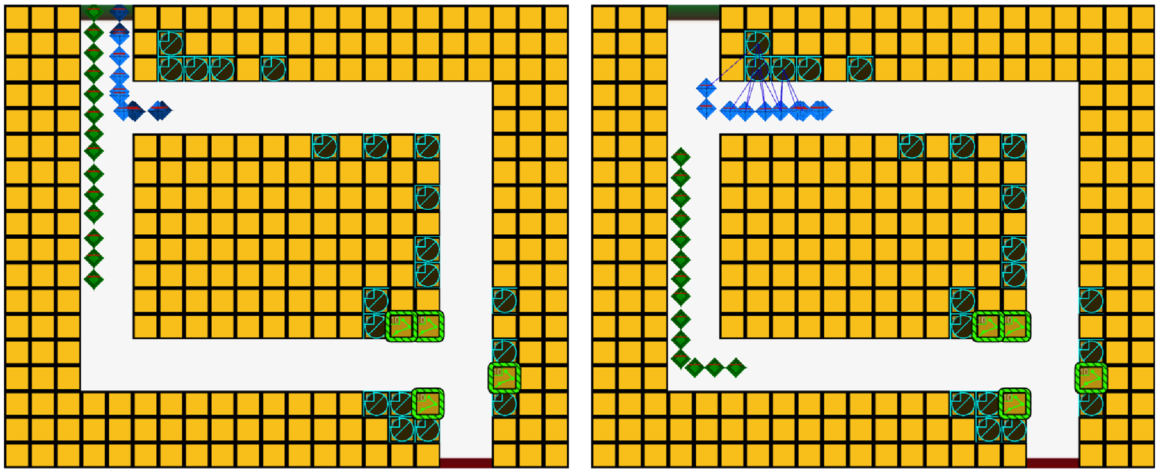
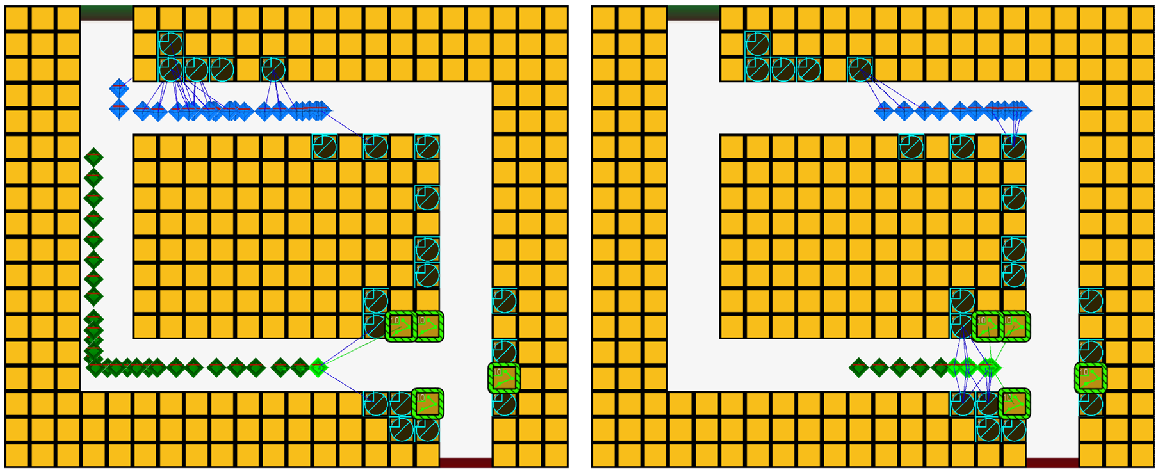
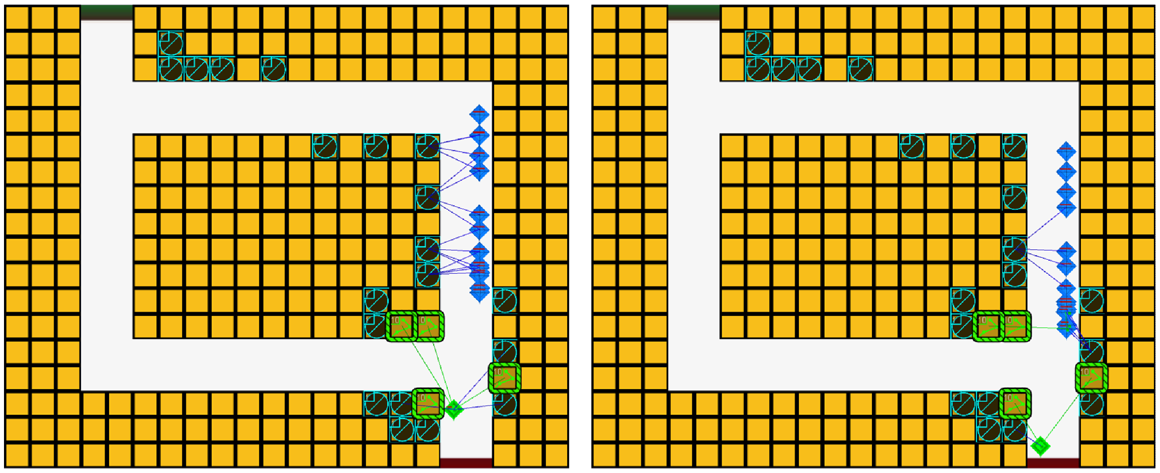
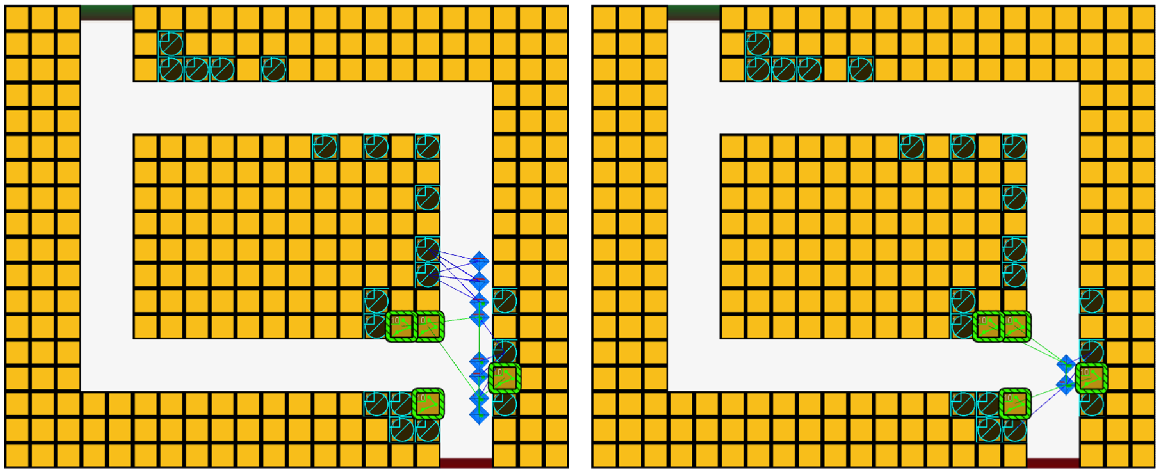


Figure F.29. Execution of Differential Slowing in Do the Splits. LEFT: The area covered by the DIFFERENTIAL SLOWING towers. RIGHT: The green offense tower range and blue slowing tower range are precisely lined at their exits from the left side, allowing the SLOW IN RANGE towers to slow the left/green line creeps the exact moment they enter the offense towers' range. They are not slowed earlier as that would left the right/blue differentially slowed creeps to catch up, reducing the path synchronization gap. They are not slowed later as that would cause the offense towers to have less time to attack.


The first two blue slowing towers can reach both lines of creeps but because the right/blue line is closer, only they are slowed. The left/green line pulls ahead.


The left/bottom/green line of creeps enters the kill zone. The right/top/blue line of creeps is still seven tiles away from the kill zone. This is partially because the kill zone is one tile shorter on that side and mostly because of the slowing towers along their path.


The green creeps are almost completely destroyed by the time the right/blue line of creeps enters the kill zone.


With all the green creeps destroyed, all four offense towers focus on the right/blue line of creeps.

F.3.4 Ladder

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	18.44	23.70	3.06	22.89
Range	-	9-27	20-27	0-18	16-28
Standard Deviation	-	5.58	2.08	3.97	2.81
Perfect Scores	-	0/25	0/20	-	-

Table F.19. Scores for Ladder, with Benchmarks.

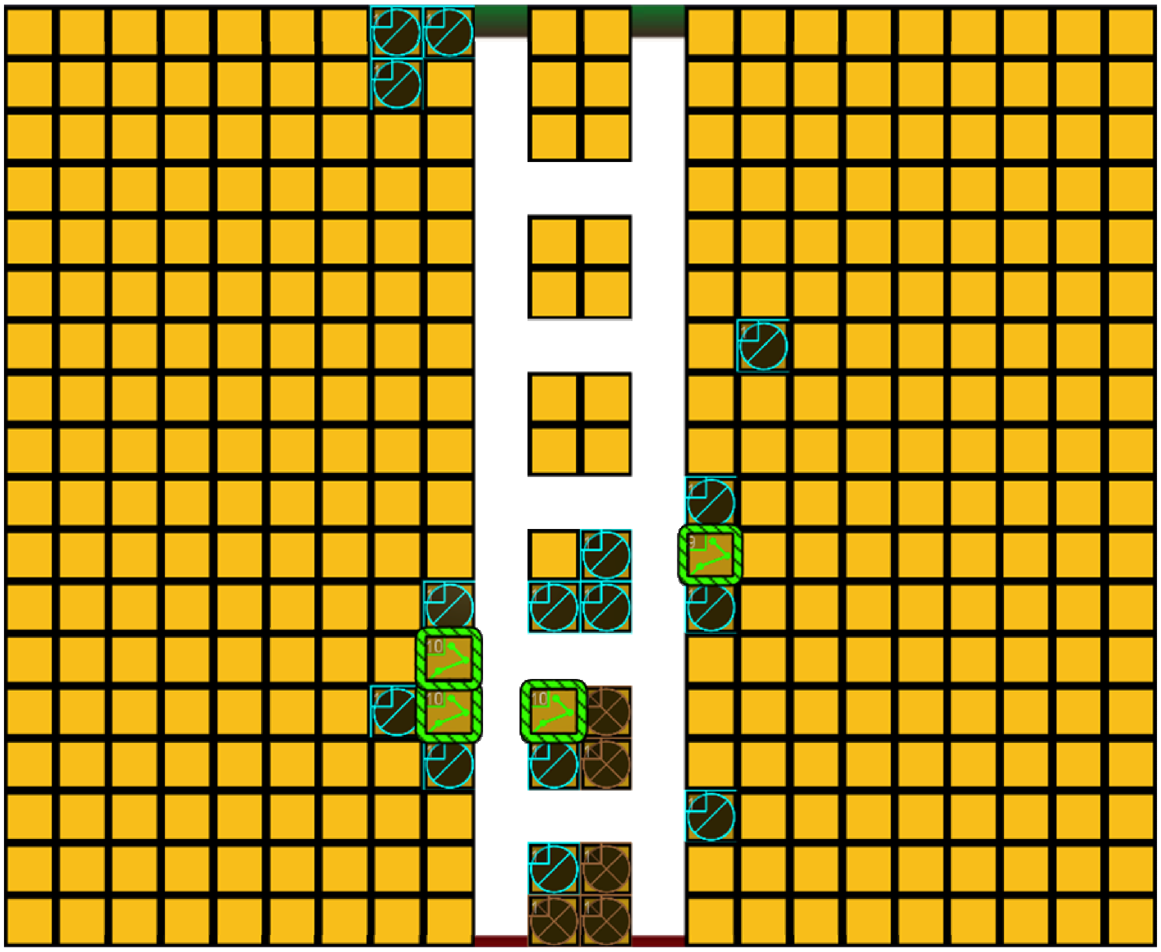


Figure F.30. Perfect Solution to Ladder. Created by author. Slowing towers are placed where they only slow the left line of creeps, giving the offense towers slightly more time to focus on the right line (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING, STRATEGY TP12: ONE BACK POSITION). Because the lines crisscross, there are few spots where DIFFERENTIAL SLOWING towers can be placed, making this strategy only minorly useful. Offense towers are placed late in the map to exploit the path synchronization gap created by DIFFERENTIAL SLOWING. Rather than being on the islands where they would have the largest usable range, they are placed on the side, losing usable range (and therefore time) but allowing them to focus on a single line of creeps (STRATEGY T9: AVOID ISLANDS). Slowing towers are used at the front of the kill zone range to give the offense towers more time to attack (STRATEGY TP1: SLOW IN RANGE). Freezing towers are used by the exit to create a cleanup area (STRATEGY P2: USE FREEZING FOR CLEANUP). A single offense tower is placed on an island, sacrificing focus but allowing it to cover the exit, catching any creeps that make it through the kill zone (STRATEGY E5: RE-USE HEAVY AS CLEANUP).

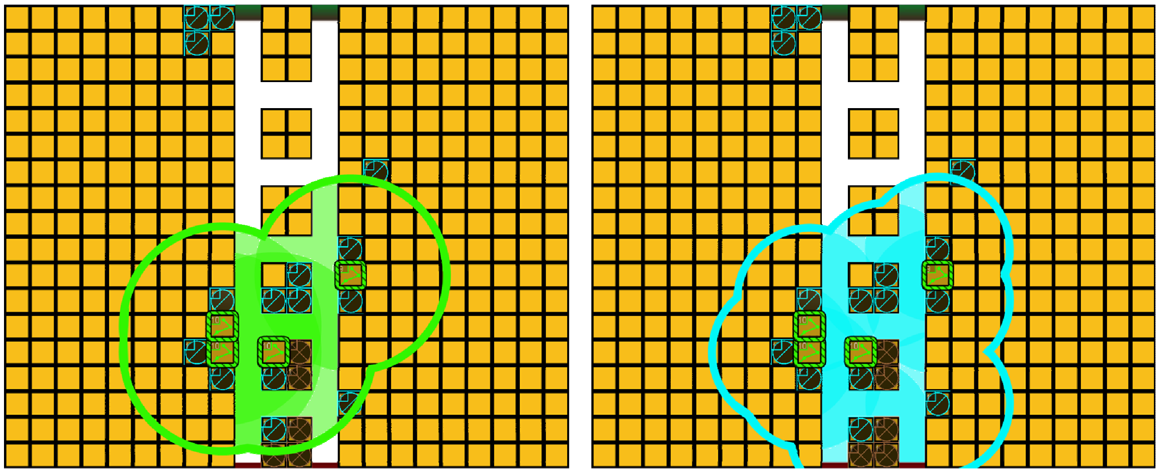


Figure F.31. Kill Zone for Ladder. LEFT: The area covered by the offense towers. The range is higher on the right, causing it to attack the right/green line of creeps to be attacked first. This is in addition to the time lag experienced by the left/blue line of creeps due to DIFFERENTIAL SLOWING. RIGHT: The area covered by the SLOW IN RANGE towers.

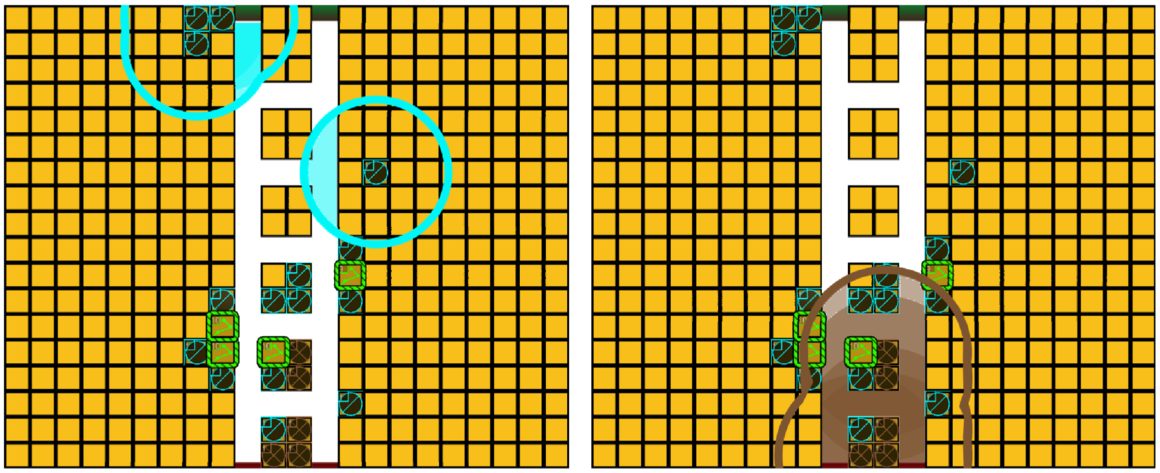
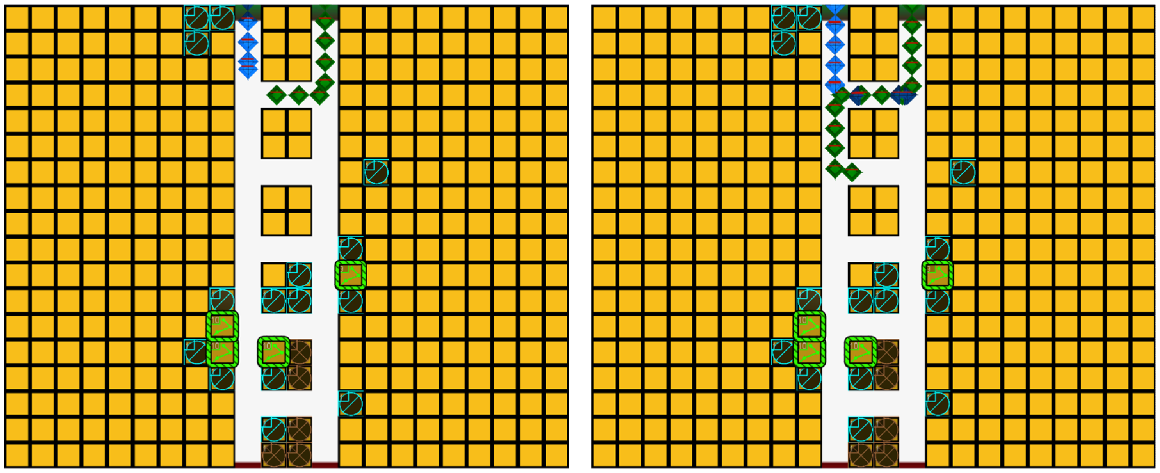
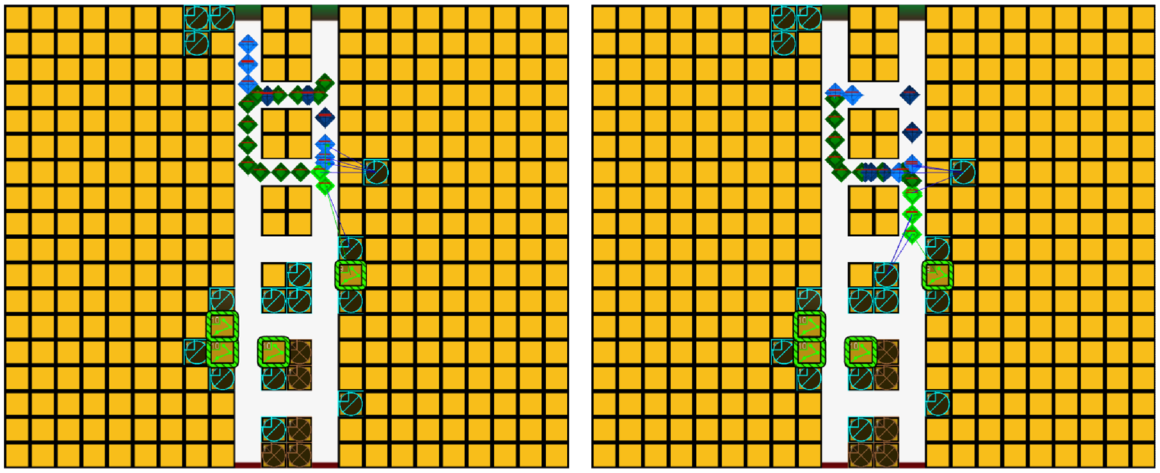


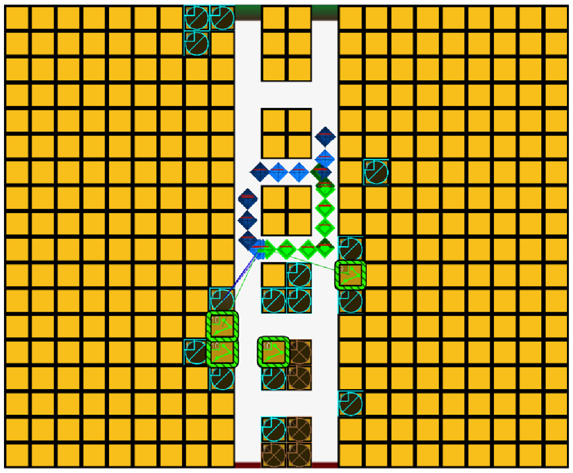
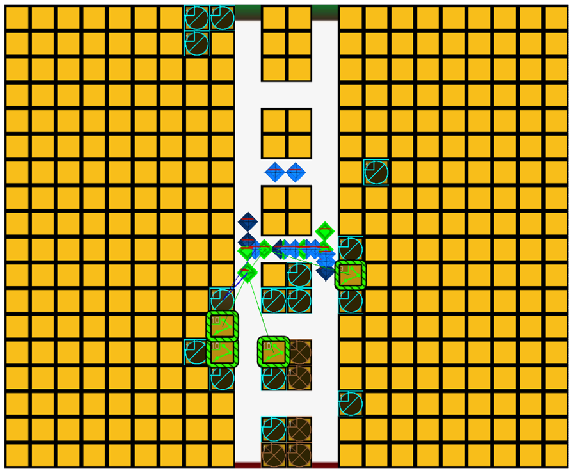
Figure F.32. Differential Slowing and Cleanup Slowing Ranges for Ladder. LEFT: The area covered by the DIFFERENTIAL SLOWING towers. The first three slowing towers only affect the left/blue line. The slowing tower on the right plays two roles. The start of its range only covers the side traversed by the left/blue line, playing a role in DIFFERENTIAL SLOWING. The bottom half of its range slows the right/green line as it enters the kill zone, playing a role in SLOW IN RANGE. RIGHT: The area covered by the cleanup area freezing towers. The five towers are able to stop five creeps and will be ineffective if any more creeps than that make it to the exit.

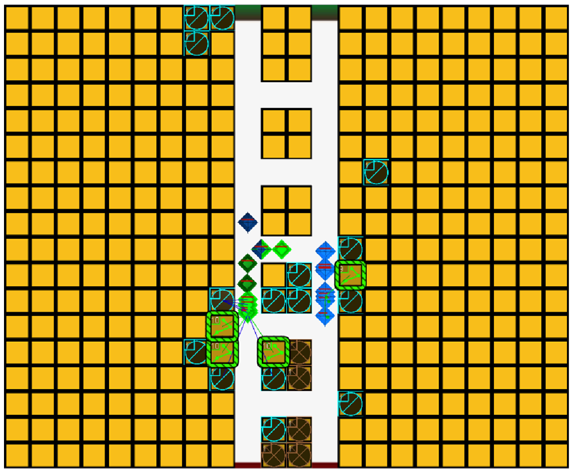
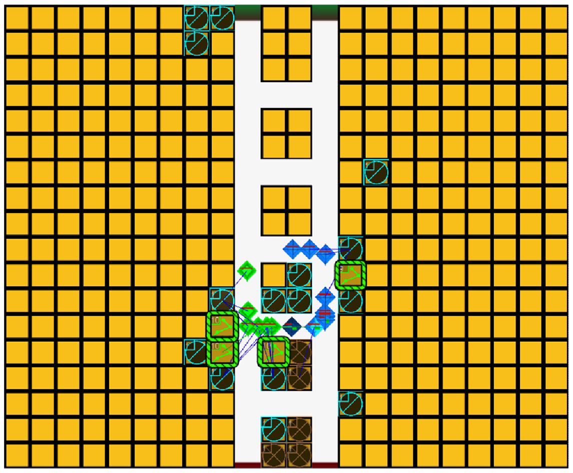


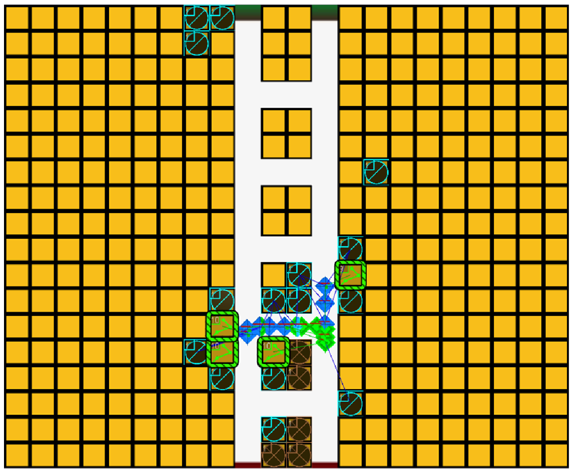
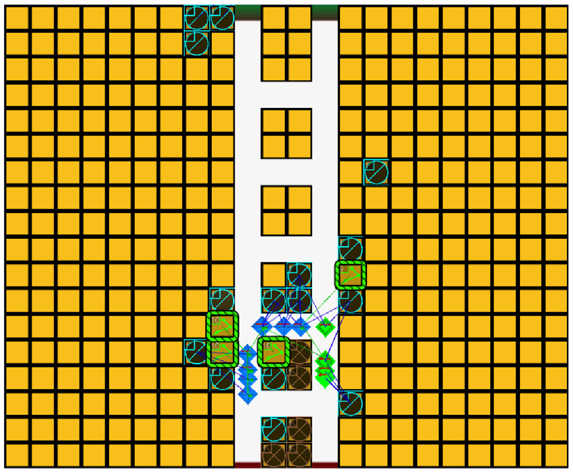
The slowing towers on the left cause the left/blue line of creeps to fall three tiles behind. This extends the kill zone's attack time by the amount of time it takes the creeps to move three tiles, increasing the offense towers' attack time by 3.75C (13%). It's not a significant contribution and results in fewer slowing towers being placed in the kill zone (the right side of the next-to-last island could use one more). Judging tradeoffs at this level (single towers with minor effects) is difficult and in many subjects appeared to be resolved with trial and error.



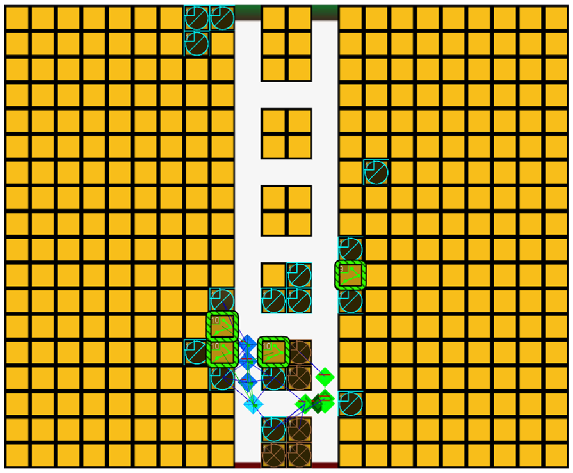
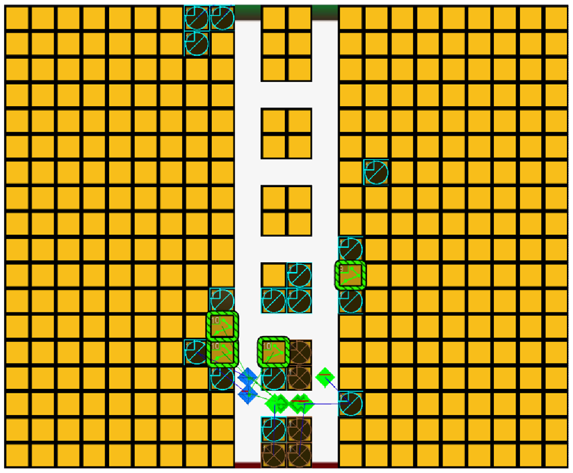
The green line of creeps enters the kill zone. The blue line is still four tiles away from entering the kill zone.



When the blue line enters the kill zone, the offense towers split their attention between the two lines of creeps. The four tile path synchronization gap disappears because the green line of creeps was slowed once it entered the kill zone. There is no obvious way of avoiding this problem on this map since the crisscrossing paths leave few areas to maintain the blue line's DIFFERENTIAL SLOWING (ruling out STRATEGY TP14: AVOID DUAL-PATH SLOWING and STRATEGY TP15: REINFORCE BETWEEN KILL ZONES). This is partially ameliorated by left-biasing the placement of the offense towers, causing the blue line to enter the core of the kill zone later.

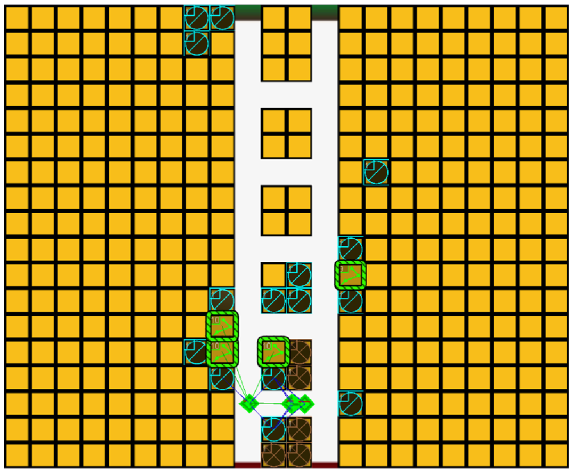
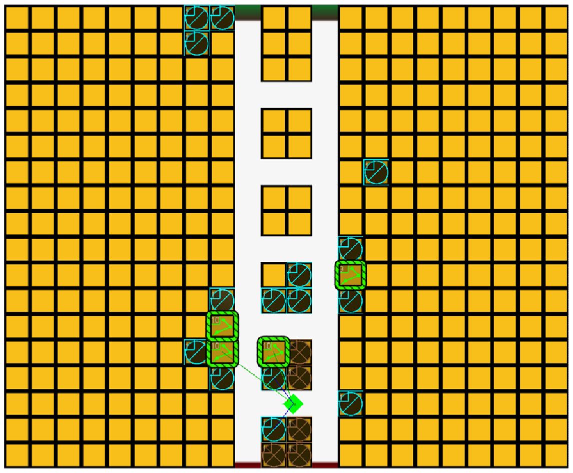


The green line enters the core area of the kill zone, where they are closer to three of the four offense towers than the other line. The blue line is closer to the fourth offense tower.

The blue line enters the core area of the kill zone as the green line exits. RIGHT: Eight green creeps have been destroyed, six remain. LEFT: Five green creeps and eight blue creeps remain. The green creeps fall behind for the first time as they enter the range of the freezing towers.

The core offense towers finish off the blue creeps while the green creeps slowly move around the freezing towers.



The heavily wounded green creeps move close to the core offense towers just as the blue creeps are destroyed. With five freezing towers holding them in place, the three offense towers are able to finish off the remaining creeps.

F.3.5 No Left Turns

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	-	15.55	4.74	13.91
Range	-	-	11-28	0-14	8-21
Standard Deviation	-	-	4.50	4.31	2.46
Perfect Scores	-	-	1/20	-	-

Table F.20. Scores for No Left Turns, with Benchmarks.

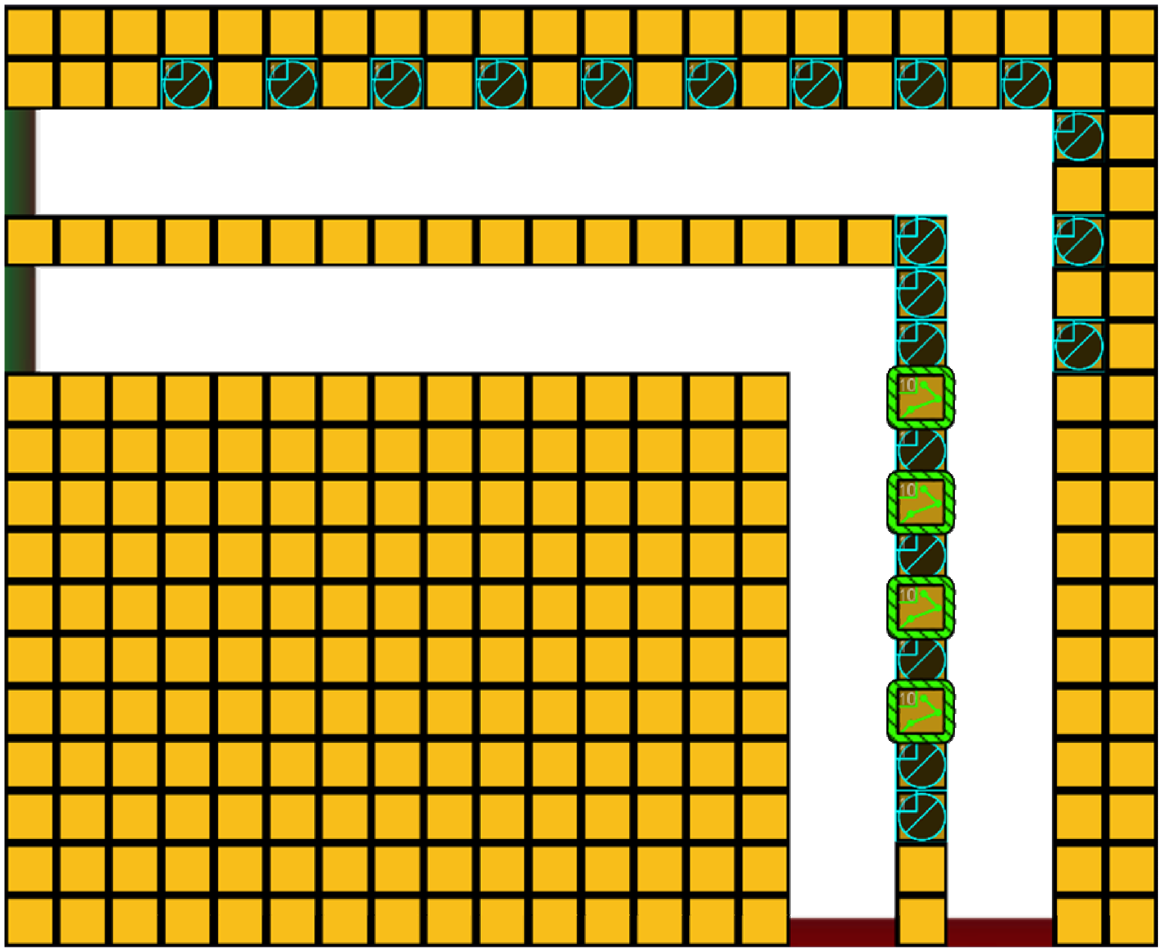


Figure F.33. Perfect Solution to No Left Turns. 3E0200's solution to No Left Turns. Offense towers are placed after the corner so that the bottom line of creeps arrives before the top line (STRATEGY TP8: EXPLOIT GEOMETRY). Slowing towers are placed where they only slow the top line of creeps, further delaying the top line and giving the offense towers more time to focus on the bottom line (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). Slowing towers slow both lines of creeps once they enter the kill zone (STRATEGY TP1: SLOW IN RANGE).

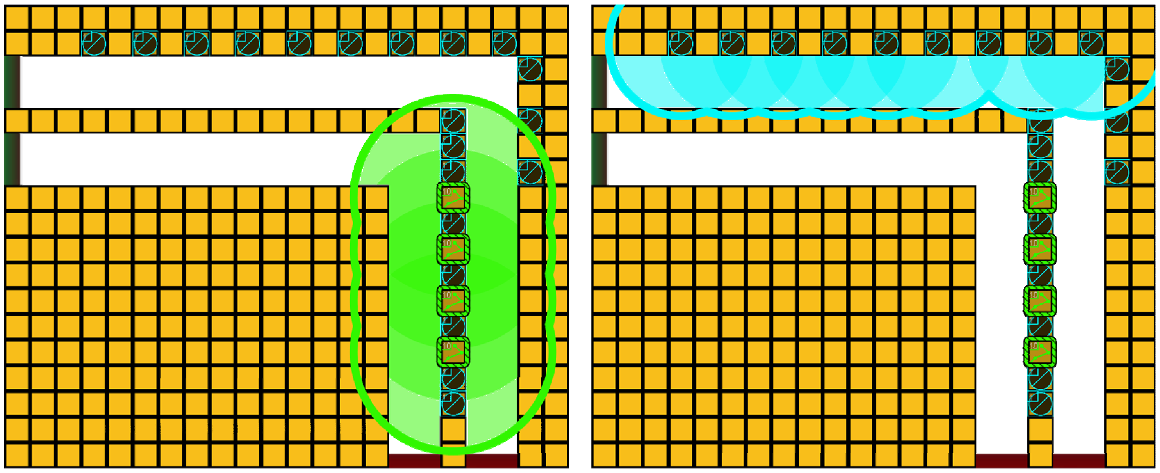
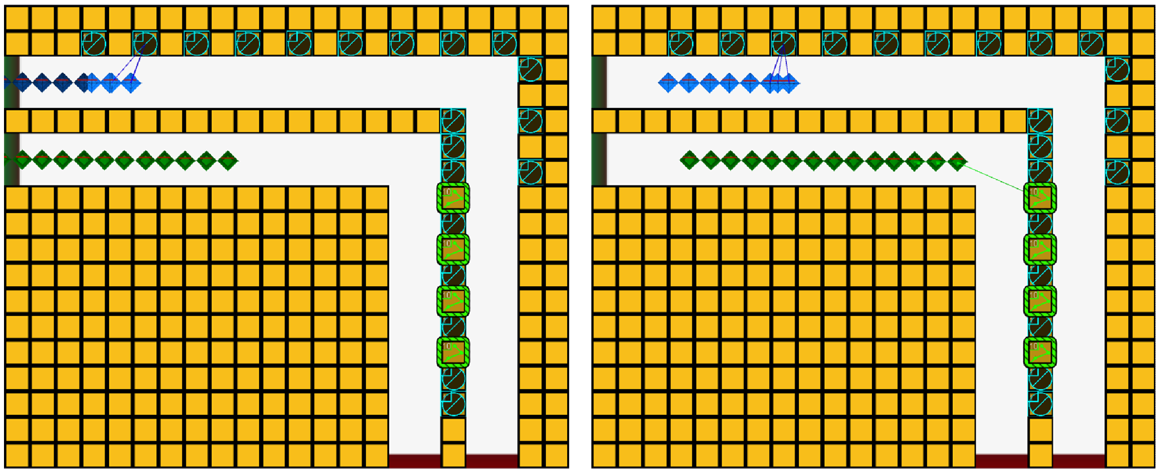
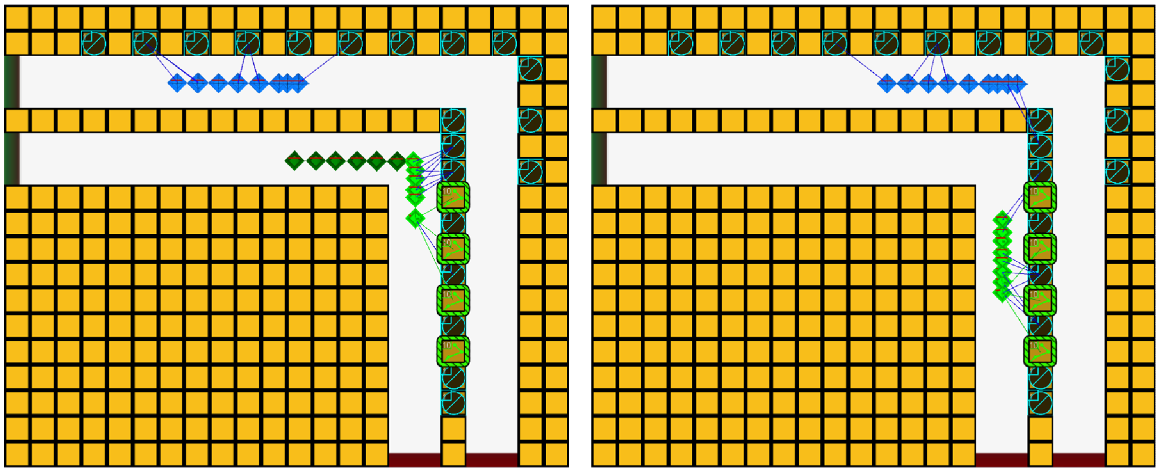


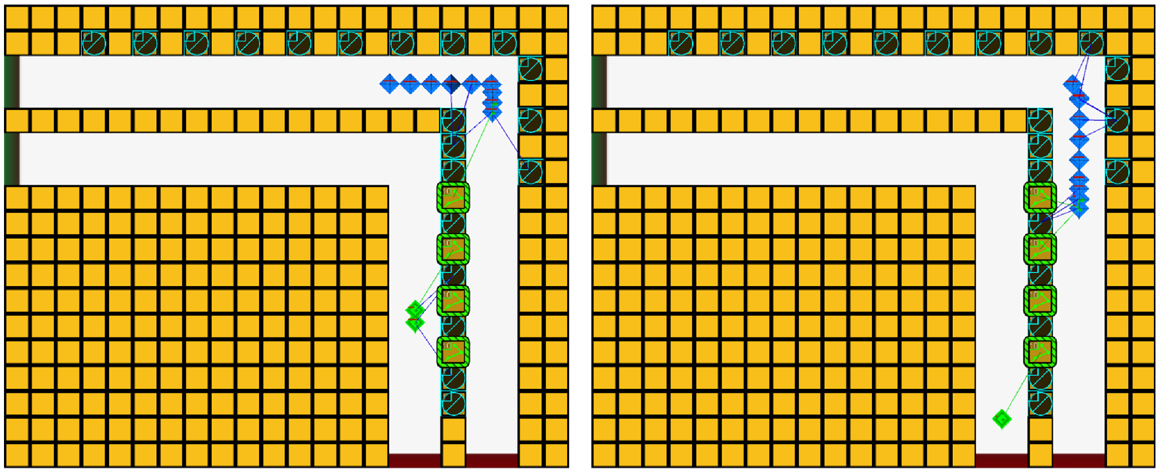
Figure F.34. Kill Zone and Differntial Slowing Zone for No Left Turns. LEFT: The area covered by the offense towers. RIGHT: The area covered by the DIFFERENTIAL SLOWING towers.



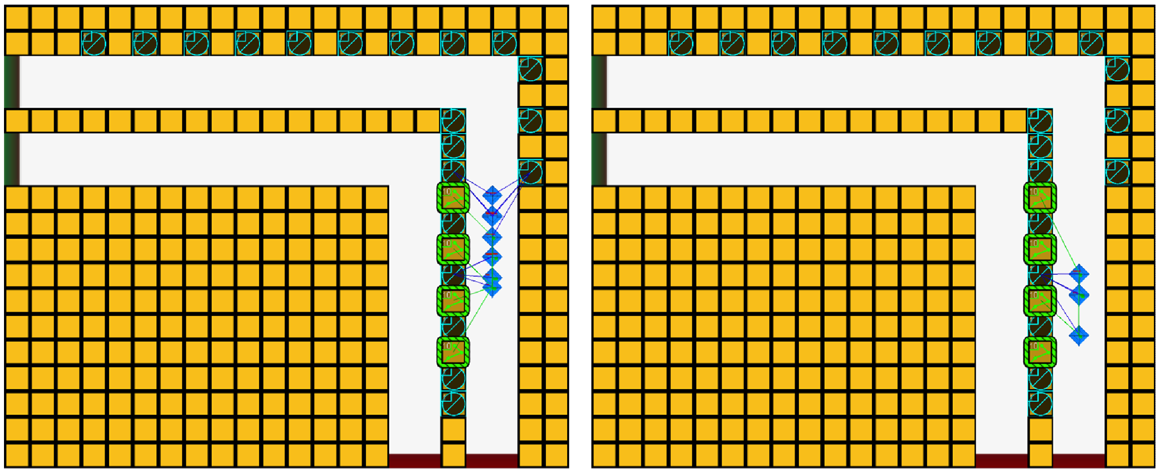
The top slowing towers can only reach the top/blue line of creeps. The bottom/green line of creeps pulls ahead. DIFFERENTIAL SLOWING needs a lot of time and/or space to separate the two lines. While the paths on this map are below average in length (28/34 vs. 60), the path from the entrance to the kill zone is long enough to completely separate the two lines.



LEFT: The bottom/green line of creeps move into the kill zone while the top/blue line is still moving through the top area. The two lines are almost completely separated. RIGHT: The entire bottom/green line of creeps is in the kill zone while no part of the top/blue line has entered.



The left/green line of creeps is in the kill zone alone, allowing the offense towers to focus all their fire on them. The top/blue line of creeps enters the kill zone just as the towers finish with the left/green line.



The right/blue line of creeps moves into the kill zone. With no green creeps to distract it, the towers can focus on the right/blue creeps.

F.3.6 Pathways

	Exp. 1	Exp. 2	Exp. 3	Random	Path Adjacent
Average Score	-	15.60	19.10	9.82	18.28
Range	-	6-28	10-28	0-21	11-25
Standard Deviation	-	5.99	5.92	4.60	3.37
Perfect Scores	-	2/25	3/20	-	-

Table F.21. Scores for Pathways, with Benchmarks.

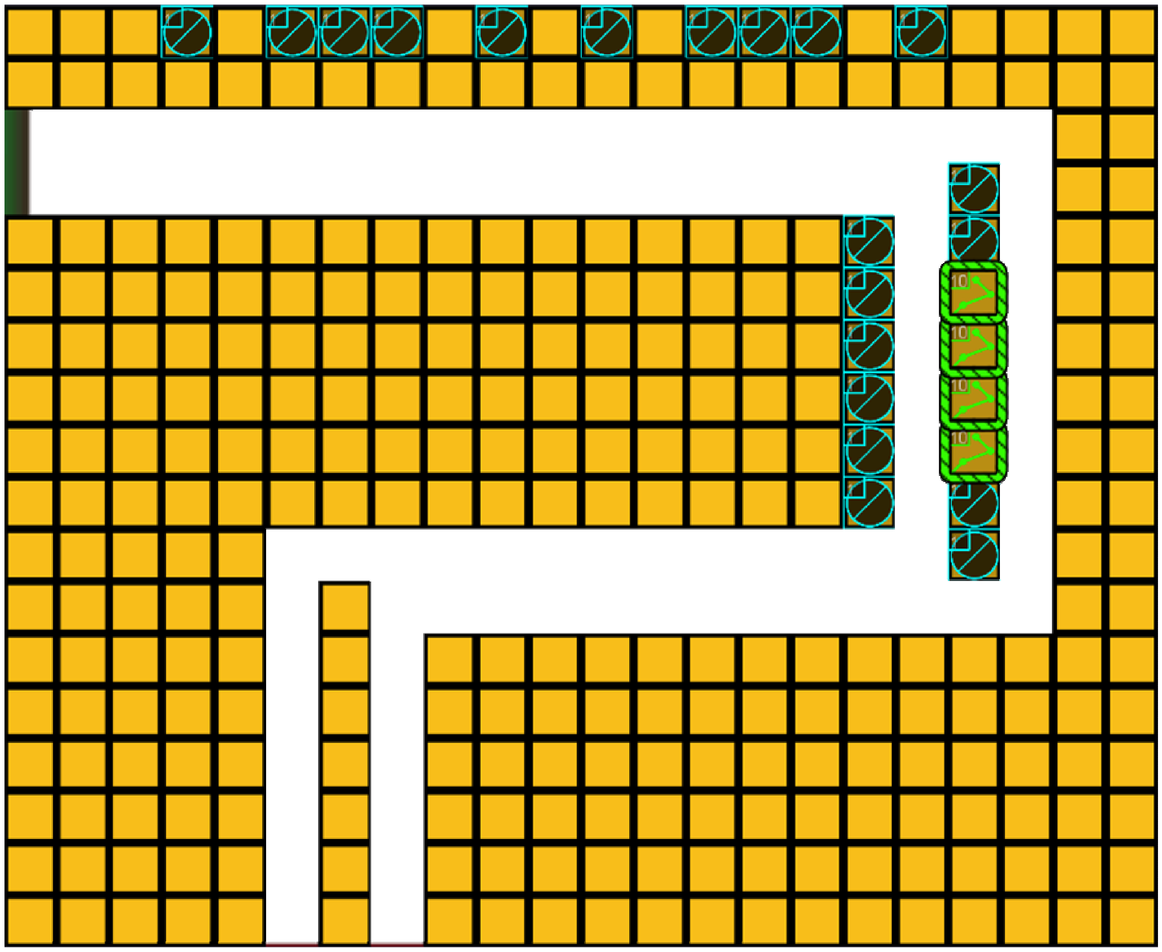


Figure F.35. Perfect Solution to Pathways. 3E0700's solution to Pathways. Offense towers are placed after the corner so that the bottom line of creeps arrives before the top line (STRATEGY TP8: EXPLOIT GEOMETRY). The top line of slowing towers are placed one tile away from the path so that they can only slow the top line of creeps (STRATEGY TP12: ONE BACK POSITION), further delaying the top line and giving the offense towers more time to focus on the bottom line (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING). The creeps are slowed once they enter the kill zone (STRATEGY TP1: SLOW IN RANGE).

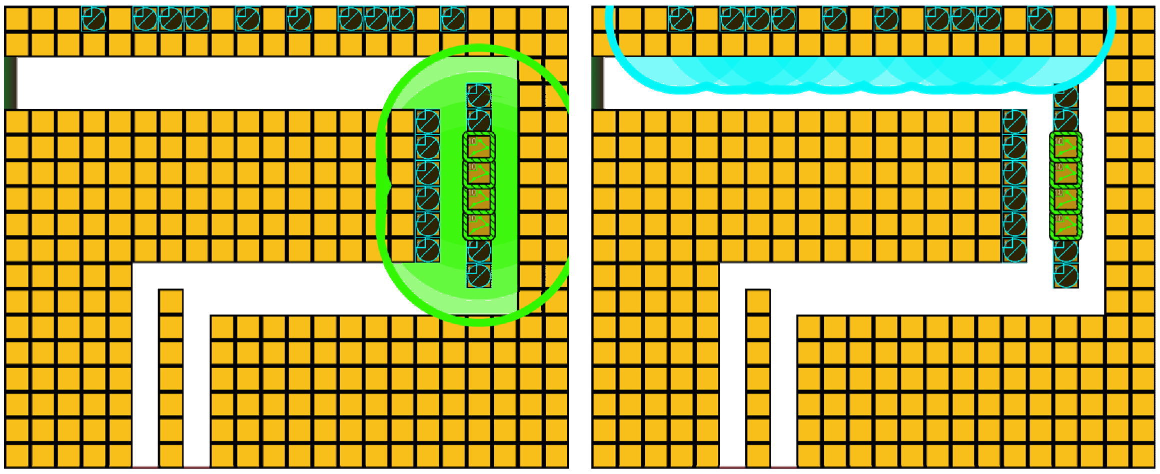
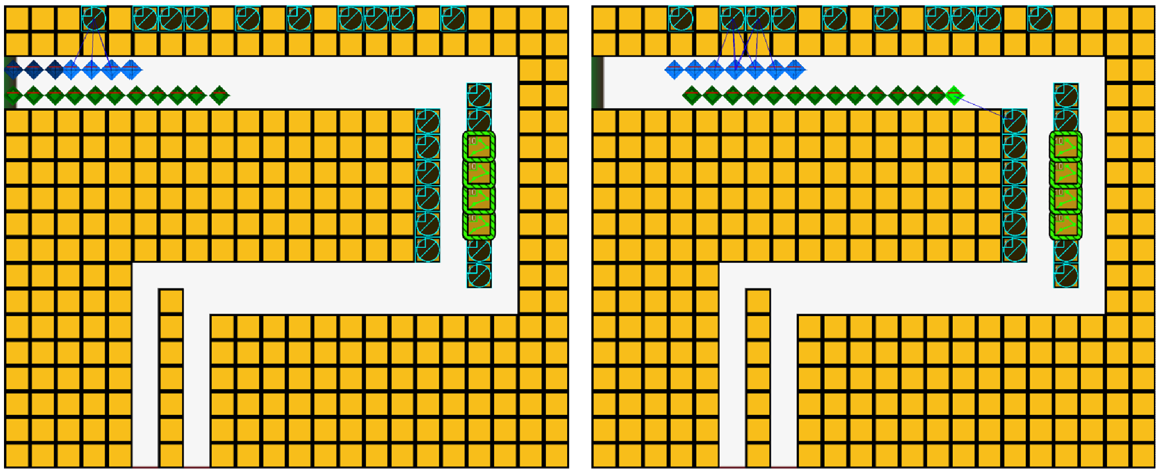
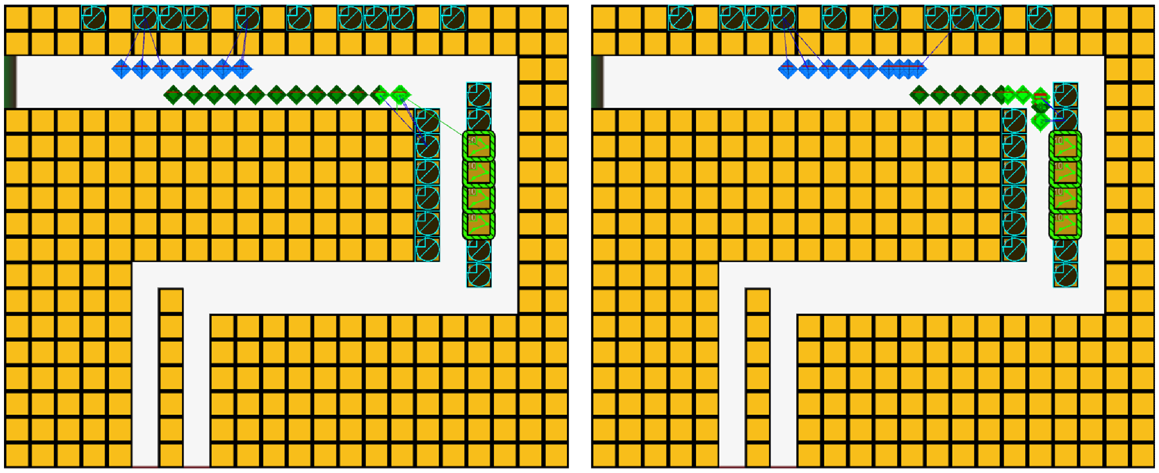
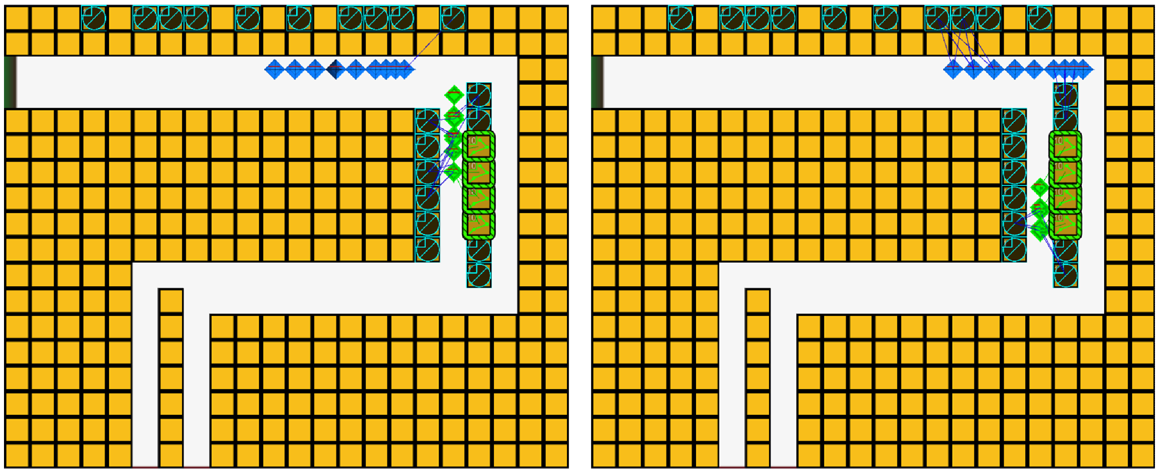


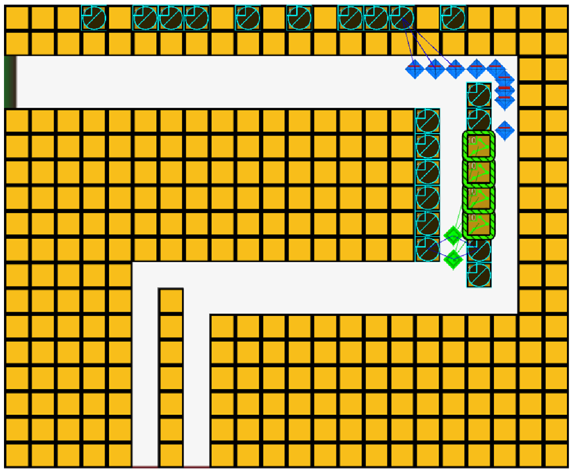
Figure F.36. Kill Zone and Differential Slowing Zones for Pathways. LEFT: The area covered by the offense towers. RIGHT: The area covered by the DIFFERENTIAL SLOWING towers.

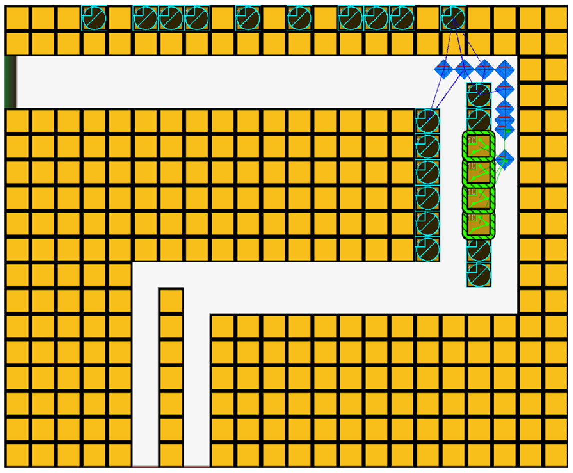


The top slowing towers can only reach the top/blue line of creeps. The bottom/green line of creeps pulls ahead.

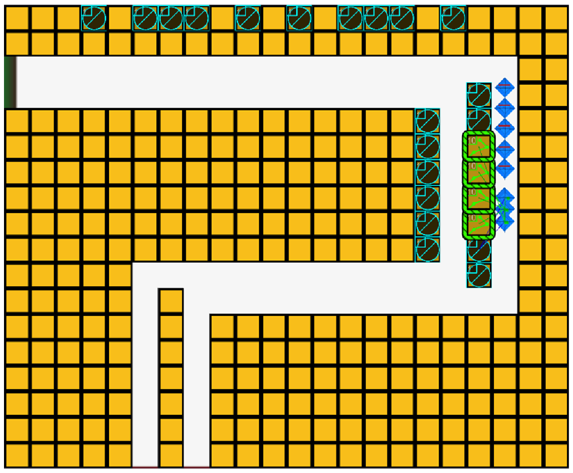

The bottom/green line of creeps move into the kill zone while the top/blue line is still moving through the top area.

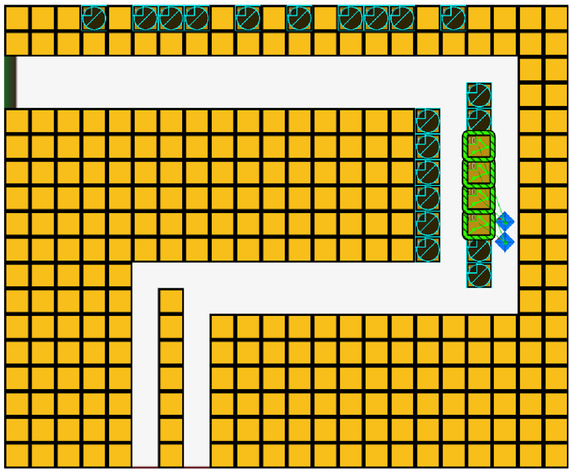

All of the left/green line of creeps is in the kill zone. None of the top/blue line of creeps is, allowing all of the offense towers to focus on the left/green line of creeps.





The top/blue line of creeps enters the kill zone just as the towers finish with the left/green line.





The right/blue line of creeps moves into the kill zone. With no green creeps to distract it, the towers can focus on the right/blue creeps.

In this section we have tried to show solutions that used the same set of towers as subjects used in Exp. three. This is not meant to imply that this set of towers is the best or only set that would work. What follows are three other solutions that earn perfect scores using different sets of towers.

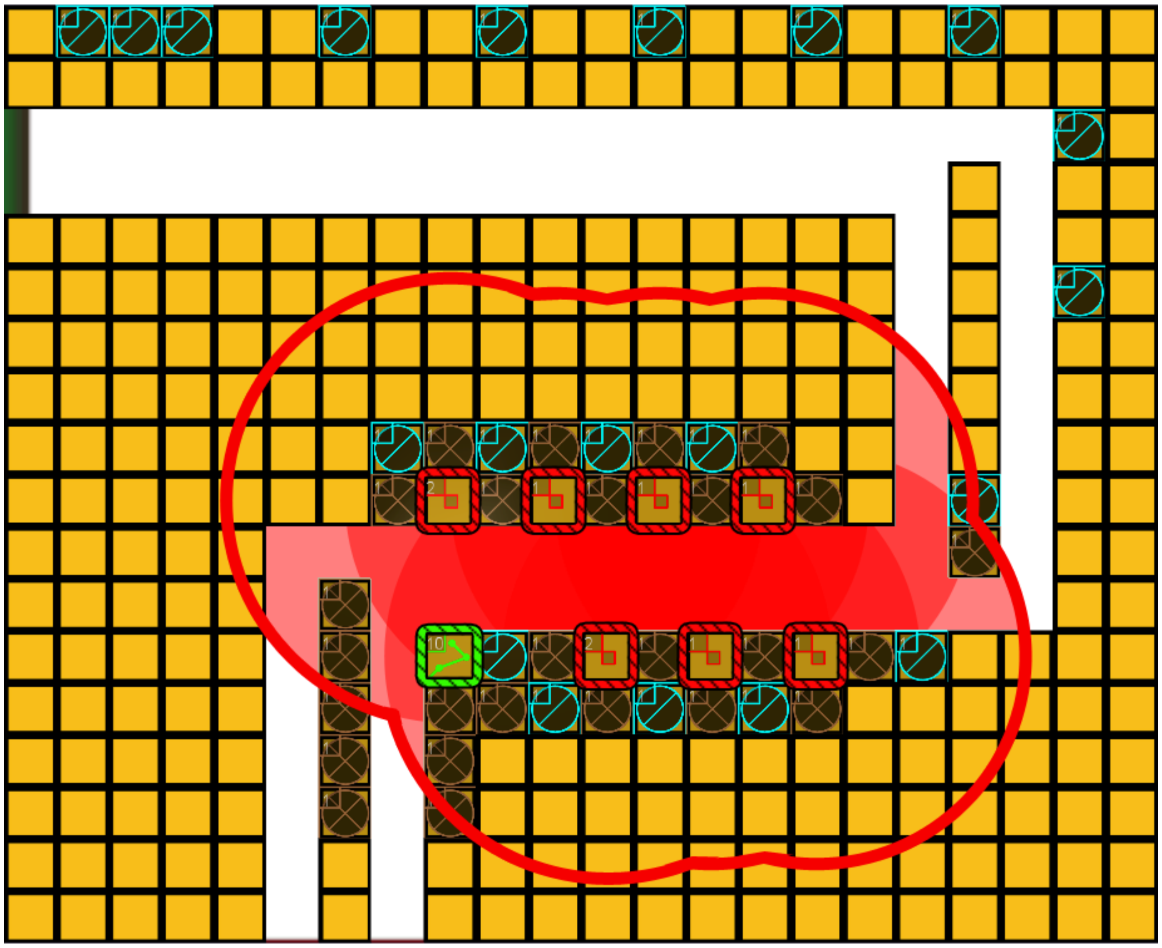


Figure F.37. Perfect Solution to Pathways. Alternate Example 1. Red3 towers do a lot of damage and have a large range but are much slower than the green towers. The additional range isn't very valuable here because the center wall is so thick that the tower has a hard time covering both sides. Because red towers are slow, additional range isn't very helpful and upgraded towers do less damage per dollar than non upgraded towers, eight level 1 and 2 Red3 towers are used (STRATEGY E12: USE MANY, NON-UPGRADED TOWERS). Even at level 1, Red3 towers do a lot of damage and risk significant overage. In addition to wasting power, it wastes a shot, preventing the tower from shooting for two seconds. To address this, two things are done: towers are not upgraded (STRATEGY E13: USE LOW POWER TOWERS) and a fast Green3 supporting tower is used to finish wounded creeps (STRATEGY E15: USE SUPPORTING OFFENSE). To give the support tower more time to cleanup wounded creeps, 16 freezing towers stop creeps in its range (STRATEGY P2: USE FREEZING FOR CLEANUP), enough power to completely hold five creeps permanently (STRATEGY TP2: SWAMP). To give the primary offense towers more time, creeps are slowed or frozen when they are in range (STRATEGY TP1: SLOW IN RANGE), the kill zone is placed after two right turns (STRATEGY TP8: EXPLOIT GEOMETRY) and the line of creeps with the longer path is slowed (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING, STRATEGY TP12: ONE BACK POSITION).

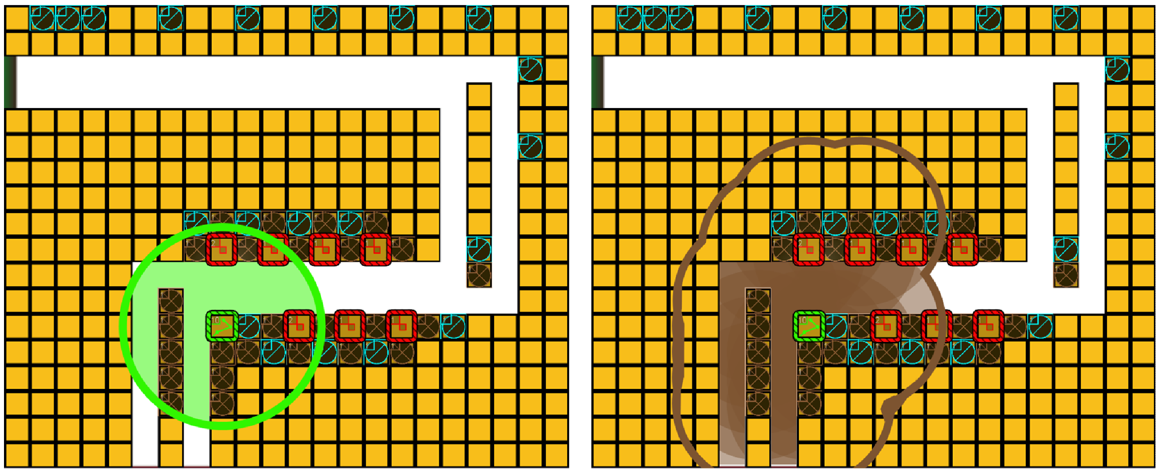


Figure F.38. Cleanup Zone for Pathways. LEFT: A level 10 Green3 fast attack tower finishes off creeps that make it more than half way through the kill zone, freeing up the stronger Red3 towers to switch to healthier targets. RIGHT: The Green3 support tower is, in turn, supported by a large number of freezing towers, forming a mini-swamp.

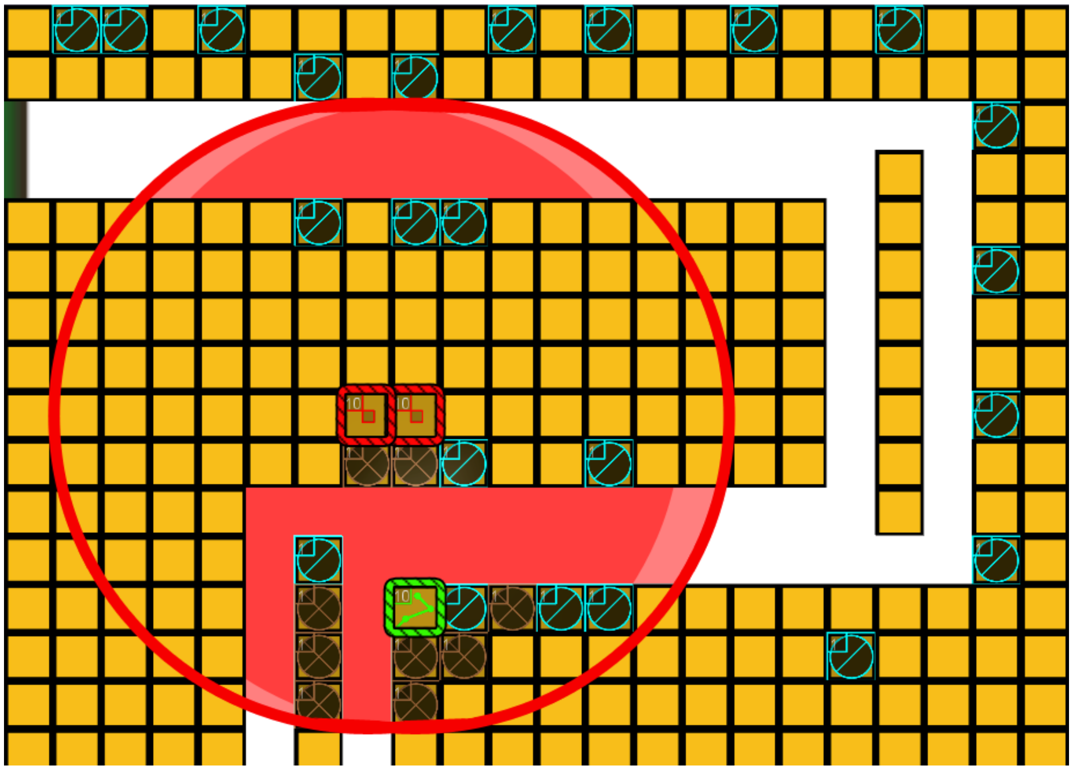


Figure F.39. Perfect Solution to Pathways. Alternate Example 2. A leveled up Red3 tower is the only tower that can reach both sides of the center wall. To these towers, this center wall is a U-turn. U-Turns form a single large attack window, leading to significant problems with attack window decay, especially for slow towers such as the Red3. To handle this, the two red offense towers are moved back from the U-turn (STRATEGY TP5: TEMPORAL ATTACK WINDOW SEPARATION). There is an opportunity to separate the two lines of creeps (STRATEGY TP8: EXPLOIT GEOMETRY, STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING, STRATEGY TP12: ONE BACK POSITION) but it is not as significant as the opportunity for other towers since the Red3 towers' first attack window is before the lines split. Leaked creeps are handled by a cleanup area (STRATEGY P1: USE CLEANUP TOWERS, STRATEGY P2: USE FREEZING FOR CLEANUP), although the primary towers are able to help as their range is able to reach almost to the exit (STRATEGY E5: RE-USE HEAVY AS CLEANUP) and they are closer to the exit than the path start (STRATEGY T8: BIAS TO EXIT). The cleanup offense tower is placed on a corner to maximize the area it covers (STRATEGY SP3: MAXIMUM USABLE RANGE – TRAFFIC VOLUME). Creeps are slowed while in the primary attack towers' range (STRATEGY TP1: SLOW IN RANGE), with slowing towers placed to make sure they don't slow outside of the attack range (STRATEGY E9: DON'T SLOW BEFORE KILL ZONE) and don't allow the creeps to move through the attack range un-slowed (STRATEGY E10: SLOW AT ENTRANCE TO KILL ZONE). Red3 towers are quite expensive. After setting up DIFFERENTIAL SLOWING and a cleanup area, not enough money was left to slow the attack range of both Red3 towers independently. To resolve this, the Red3 towers were placed next to each other so that they can share slowing towers (STRATEGY E3: CONCENTRATE TO SHARE SLOWING).

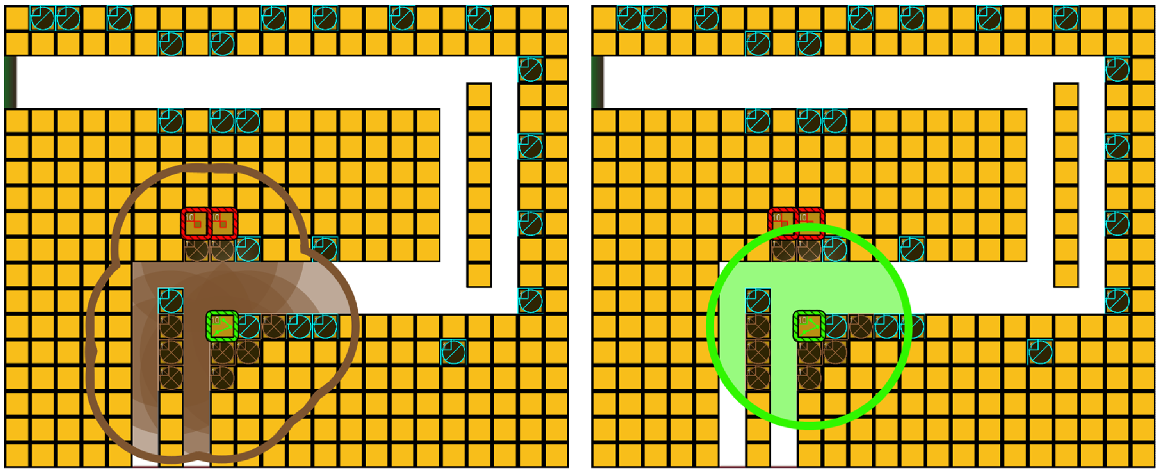


Figure F.40. Cleanup Zone for Pathways. LEFT: A level 10 Green3 fast attack tower cleans up off leaked creeps. RIGHT: The cleanup area contains a number of freezing towers.

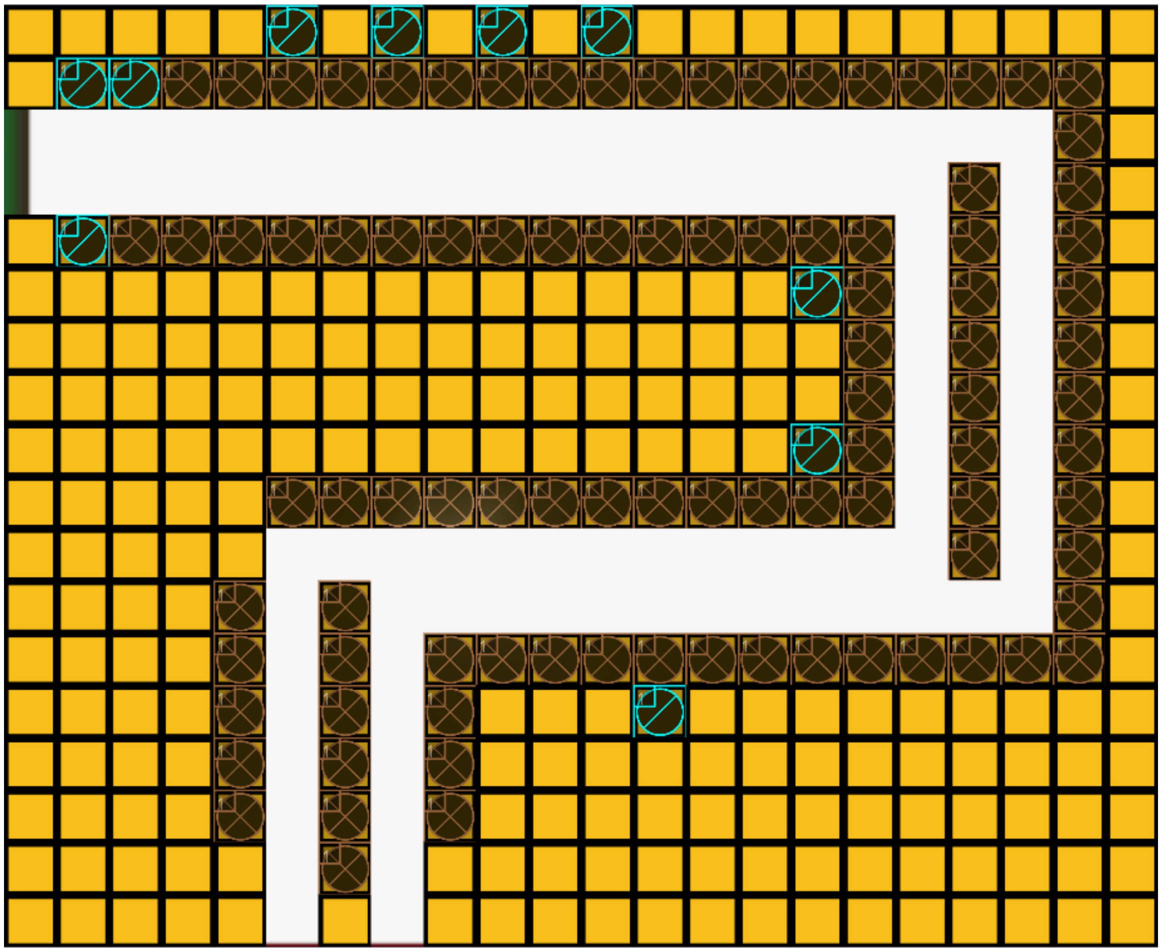


Figure F.41. Perfect Solution to Pathways. Alternate Example 3. SWAMP ATTACK (STRATEGY TP2: SWAMP) assisted with DIFFERENTIAL SLOWING (STRATEGY TP9: PERMANENT LIMITED VIEW DIFFERENTIAL SLOWING, STRATEGY TP12: ONE BACK POSITION).