Improving Communication in Networked Systems using Mobile Robots

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Ahmet Onur Tekdas

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Prof. Ibrahim Volkan Isler

June, 2011

Acknowledgements

I would like to express my sincere gratitude for all people who have helped and inspired me throughout my Ph.D. study.

I am heartily thankful to my adviser, Prof. Volkan Isler, for giving me the opportunity to pursue my Ph.D. degree with him. His inspiration and endless support from the first day enabled me to successfully finish this thesis.

I would like to thank my committee members Prof. Janardan, Prof. Roumeliotis and Prof. Santosa for their invaluable feedbacks. My special thanks go to Prof. Janardan. It has been a privilege working with him. I would also like to thank Georganne Tolaas for solving administrative issues lightning-fast for me.

It would be impossible to finish this thesis without financial support. I am indebted to National Science Foundation for supporting this work with the grants: 0907658, 0917676 and 0936710.

I have had the honor to meet many wonderful people in Robotics Sensor Networks Lab. I would like to thank Pratap Tokekar for inspiring me with his robotics expertise. I am thankful to Deepak Bhadauria and Patrick Plonski for their contributions in this thesis. My special thanks go to “the Dude” - Nikhil Karnad who is like a brother to me. Dude, I am sincerely thankful for your help and friendship.

My deepest gratitude goes to my parents for their endless love and continuous support. I would not be the person I am today without their help. TR: [Annecim, babacim, beni ben yapan sizin sonsuz sevginiz ve desteginiz. Herseyi size borcluyum.] To my brother, you are the most beloved person in my life since the minute you were born.

To my love and my soul-mate ... Everything makes sense with you and is meaningless without you. TR: [Dadlum, askim, bitanem bir gunes gibi dogdun karanligima, es oldun yalniz ruhuma, iyi ki yanimdasin.]

Dedication

To my beloved parents and brother ...

Bu tezi sevgili anne ve babama, ve biricik kardesime adiyorum.

Abstract

Providing network communication in large, complex environments is an important task with applications to maintaining connectivity with mobile users, environmental monitoring, emergency response, search and rescue, etc. Traditional approaches accomplish this task by deploying a static wireless network over the entire environment. However, this solution becomes cost ineffective when the area to be covered is large.

Recent advances in robotics technology and research have made it possible to build low-cost, robust mobile robots that can autonomously navigate in complex environments. Thanks to these advancements, it is now feasible to use robots as mobile network nodes in place of large static network deployments. However, in order to achieve good performance with a small number of robots, it is crucial to design efficient algorithms for planning the robots' paths. In this dissertation, we study the use of mobile robots in two specific networking applications.

In the first application, we use mobile robots to provide communication between end-points that require a persistent connection in a large, complex environment. For instance, imagine that a mobile user in a large environment requests connectivity to a static base station. Since the service area of wireless routers is limited by their initial deployment locations, a static wireless network deployment requires many routers to fully cover the entire region. Alternatively, this drawback can be overcome by using a small number of robots as intermediate mobile routers between the user and the base station which adaptively maintain connectivity according to the user's movement.

In the second application, we seek the use of mobile robots in delay-tolerant networks where a small delay in data transfer is acceptable. One such application is environmental monitoring where scientists collect statistical data such as soil temperature. The most crucial problem in this application is to gather the data from sensors which may be sparsely deployed over a large environment. We can avoid the inefficient use of intermediate relay nodes for data transfer by using mobile robots to autonomously collect the data from sensors. Since a small delay is tolerable, using a few robotic data carriers becomes an appealing solution.

Our contributions in this dissertation are two-fold: on the theoretical front, we present path-planning algorithms with provable performance guarantees. First, we study the problem of maintaining the connectivity of a mobile end-point to a static end-point by using the minimum number of mobile routers. Second, we present solutions for creating a communication bridge between two static end-points by minimizing the number of robots and their movements. Third, we study the problem of finding robot paths so that robots collect data from sensors as quickly as possible. Lastly, we present strategies for robots which act as mobile sensors to efficiently monitor an environment. On the systems front, we implement our algorithms using mobile robots and demonstrate their practical feasibility through extensive experiments.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The robotics revolution started in the 1960s with the invention of the first robotic arm [8]. Since then, robotics has become the engine of industrial development. Motivated by its use in factories, early robotics research focused on fundamental problems such as kinematics, manipulation, grasping, etc. In the new millennium, a new era in robotics has started: *mobile robotics*. The advancements in mobile robotics were first driven by the requirements of space missions and military applications. The Mars exploration rover Spirit [9] developed by NASA, the bomb disposal robot Packbot [10] developed by iRobot and the Humanoid robot Asimo [11] developed by Honda are the world-renowned mobile robots in this category.

Later advances in mobile robotics were triggered by the necessity for low-cost yet robust mobile platforms. Especially, the growing demand for domestic robots forced the producers to develop robust robotic platforms with small cost. As an example, the iRobot Roomba vacuum cleaning robot (the first commercially successful domestic robot) has sold 6 million units at the time this thesis was written [12]. While the capabilities of these robots become more complex everyday, their costs are getting lower. Moreover, they are robust enough that the manufactures offer a standard one year warranty. Hence, now it is possible to buy off-the-shelf, low-cost and robust mobile robots that can accomplish complex tasks. Some of the standard platforms used extensively in robotics research and education are iRobot Create (programmable version of Roomba), MobileRobots Pioneer [13] and WillowGarage PR2 [14].

Autonomous navigation has been one of the most fundamental problems in robotics.

In pursuit of an accurate navigation system, a robot must be able to precisely estimate its own state (position and orientation) with respect to a fixed coordinate frame. This is achieved by various sensors such as encoders, camera, laser, etc. mounted on the robot. However, sensory measurements are not perfect which results uncertainty in robot's state estimation. Hence, a robot must fuse sensor measurements to have a better estimate of its own position. With the recent progress in state estimation research and sensor technology, it has become possible to build robots that can navigate robustly in both indoor and outdoor environments. In recent years, Simultaneous Localization and Mapping (SLAM) [15] has become a standard for indoor navigation. Moreover, the DARPA Grand Challenge 2007 showed that it is feasible to build driverless cars that can navigate autonomously in urban areas [16]. Recently, both GM and Google have started a driverless car project and it is anticipated to have the first car on road by 2018.

Since it is possible to build low-cost and robust robots with autonomous navigation capabilities, robotics researchers have turned their attention to potential use of mobile robots to achieve tasks that were infeasible or were not efficiently performed before. One such potential use of mobile robots is to improve communication in networked systems.

In this thesis, we explore the use of mobile robots to improve the communication in two types of network applications. In the first application, we explore the use of controlled mobility to provide communication between end-points which requires persistent connection. In the second application, we study the use of robots in *delay-tolerant* networks where a small delay in transferring data is tolerable. One application of such systems is *environmental monitoring* where scientists deploy a network of sensors over an environment to collect information about the environment without the need for real-time data.

In the next section, we present an overview of these applications where a few robots can provide appealing solutions to improve the communication in the network. In Section 1.2, we present underlying path-planning problems in order to achieve these improvements by using robots efficiently. For each problem, we present related work and state our contributions briefly, leaving the details to individual chapters.

1.1 Overview of Applications

In this section, we present examples where mobile robots can improve the efficiency in two types of networked systems. The first type is the *end-to-end networks* where a persistent connection between two end-points must be established. We present two applications according to the mobility of one of these end-points. The second type of networked system is the *delay-tolerant networks*. We present examples where a few mobile robots can reduce the number of sensors required or even remove the necessity for sensors in delay-tolerant networks which are used frequently in environmental monitoring applications.

1.1.1 End-to-End Network Connectivity

Consider a scenario where a single mobile user needs network connectivity. The mobile user could be a person with a laptop who needs Internet connectivity or a firefighter truck which needs continuous connectivity to a command center. The traditional approach for providing network connectivity for a mobile user is to cover the environment that the user resides in with a network of static wireless routers. Figure 1.1-Left shows an example. However, static network deployments have three main drawbacks.

First, if the environment that the user resides in is large, this solution requires many routers to fully cover the environment. This leads to a very costly solution. Second, a Local Area Network (LAN) connects to a Wide Area Network (WAN) such as Internet through access points (gateways). Hence, in the static network only the routers on the path between the user and gateway are active at a time (see Figure 1.1-Left). Since a large number of the routers are inactive in most of the time, this solution leads to inefficient use of resources. Lastly, in hostile battlefields it would be useful to extend the communication to support unmanned vehicles. However, if the underlying communication medium does not exist, deploying a network infrastructure would be impossible due to the danger of the mission.

The main reason of the inefficiency of the stationary solution is that the service provided by a static router is limited to the immediate vicinity of its placement location. Hence, adding mobility to the routers helps to overcome these drawbacks and provides a more efficient solution for network communication.

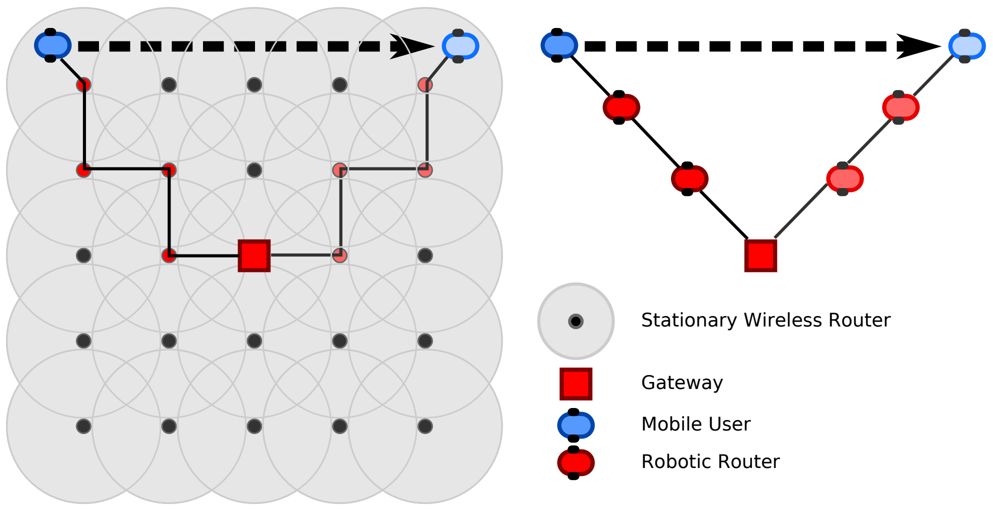


Figure 1.1: **Left:** The environment is covered with a network of stationary wireless routers where a single user communicates with an access point (gateway) through router-to-router links. Red circles show the active nodes. The next location of the mobile robot and the communication path are represented in faded colors. **Right:** A robotic router solution where the user (blue robot) is connected to the base station (red square) through intermediate robotic routers. When the user moves, the robotic routers reconfigure themselves accordingly to maintain the connectivity.

Robotic Routers

Robotic routers are mobile robots with communication and routing capabilities (a commercial robotic router is shown in Figure 1.2-Left). We can use a small number of robotic routers to create an adaptive network between the mobile user and the gateway. When the user moves, robotic routers reconfigure themselves accordingly to maintain the user's connectivity. Hence, limited resources are used more efficiently by introducing mobility. Moreover, in battlefields where deploying a static network is infeasible, robotic routers can autonomously deploy themselves in the environment without risking lives [1] .

On-demand Communication Bridge

In the previous scenario, we use mobile robots to provide connectivity between a mobile and a static node. We can also use mobile robots to create an on-demand communication between two static end-points. For instance, consider the scenario where firefighters are fighting a forest fire. A connected network between a command center and the disaster area makes it possible to exchange the global information in command center (e.g. satellite images) with the local information in the disaster area (e.g. images from disaster). This helps the command center to guide the operation more effectively and helps firefighters fight the fire more efficiently. However, in the presence of disasters, it might be irrational to expect an underlying communication medium. This communication might not exist or it might be no longer in working order. In these circumstances, we can use robots to autonomously establish a communication bridge between the command center and the area of interest. Moreover, robots can be reused for establishing connectivity between any stationary entities. For example, if the number of robots is limited and there are more multiple disaster areas, then mobile robots can be reused for providing on-demand connectivity.

[1] DARPA started the LANdroid project (see also Figure 1.2-Left) together with iRobot to build mobile robots which provide network communication for war-fighters in the battlefield.

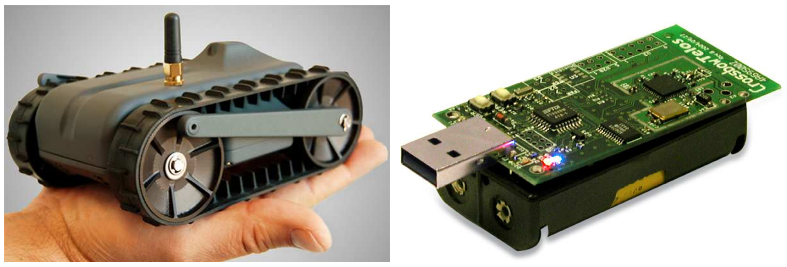


Figure 1.2: **Left:** A commercial robotic router called *LANdroid [1].* **Right:** A common wireless sensor called *telosB mote.* [2]

1.1.2 Delay-tolerant Networks

A *Wireless Sensor Network (WSN)* is a network of small and inexpensive sensor devices with wireless communication capabilities (see Figure 1.2-Right). One of the most common applications of WSN is *environmental monitoring.* In environmental monitoring, scientists deploy sensors over an environment of interest to collect statistical information (e.g. temperature, humidity, etc.) about the environment. Since the main interest is statistical information, real-time data is not necessary. Hence, we refer to such sensor network deployments as *delay-tolerant networks.* Moreover, these sensors have to be deployed over large areas and collect measurements for years to provide reliable information about geographic and temporal changes.

The main challenge in this type of deployment is gathering the collected data from the sensors. The traditional approach to achieve this task is to create a connected WSN and transfer the data over the network (see Figure 1.3-Top) via node to node links [17]. However, this solution has two main drawbacks.

First of all, in most of the environmental monitoring tasks, the environment of interest is very large and the sampling locations are far apart from each other. Hence, a large number of sensors has to be deployed to maintain the connectivity of the network which leads to a very costly solution. Moreover, deployment and maintenance of a large network could be very labor-intensive which further increases the cost of this approach.

Secondly, since sensors are mostly battery operated, their limited energy is another key challenge. The main energy consumer of a sensor is its radio, which consumes twenty times more energy than its processing unit. As a consequence, sensors spend

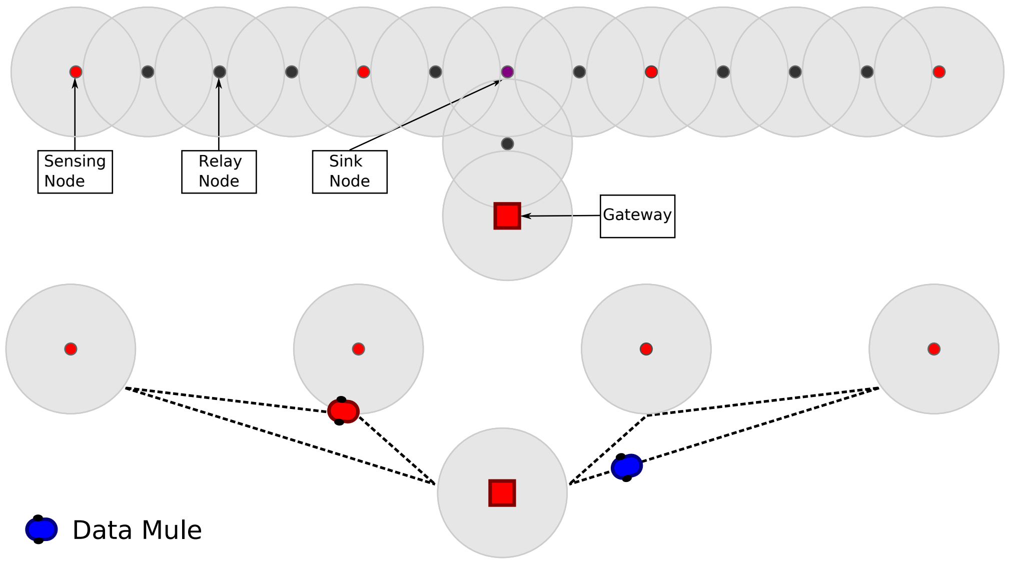


Figure 1.3: **Top:** A simple example of WSN deployment. Red circles shows the sensing nodes which collect data from the environment. Black circles show the relay nodes. The purple circle shows a special relay node which is called a sink node. In the traditional approach, the data collected from sensing nodes is transfered over relay nodes to a gateway which stores the data. **Bottom:** In an alternate method, we can use mobile robots as data mules to autonomously collect data from sensing nodes and carry the data back to the gateway.

most of their energy while forwarding the data from other sensors in addition to their own collected data. Moreover, some nodes might act like a sink node that transfer the data forwarded by multiple sensors (see Figure 1.3-Top). This causes the sink nodes to exhaust their energy source and die quickly, which might lead to loss of network connectivity.

Data Mules

An alternate to static deployment is to use robots as *data mules* [18] where robots periodically visit sensor nodes, download the collected data from them and carry the data to a center where we store the data (see Figure 1.3-Bottom). Since the aim is to collect statistical information about the environment over long period of times, the delay due to data muling is acceptable.

Data mules overcome the challenges discussed above and improve the efficiency of WSNs. Thanks to recent advances in mobile robot technology, now it is feasible to build low-cost and robust robots that can navigate autonomously and accomplish data-muling tasks. Hence, using a small number of mobile robots provides a more cost-effective solution than deploying a connected network. Moreover, since sensors directly offload their data to the robots, sensors do not need to forward the data of the other sensors anymore. Furthermore, since robots can get very close to the sensors, we can reduce the transmission power of the sensors which further improves their energy efficiency.

Multi-robot Patrol

In the previous example, scientists monitor environments with a sensor network deployed sparsely. However, in some environmental monitoring applications, it might be necessary to fully monitor the environment. For instance, if the monitoring task is to detect a potential fire in a forest, the entire forest must be monitored so that a potential fire can detected independent of where it starts. As discussed before, it would be infeasible to cover the entire forest with static sensors. On the other hand, we can use a few mobile robots (e.g. UAVs) as mobile sensors to patrol the environment to detect a potential fire. In this mobile solution, robots can monitor only a small portion of the environment at a time. Hence, a potential fire might not be detected immediately. However, we can choose the robot paths suitably to minimize the delay between the time a fire starts and the time it is detected.

1.2 Problem Statement and Contributions

In the previous section, we discussed four fundamental applications in which mobile robots can be used to improve communication in the networked systems. The most crucial problem in these applications is path-planning for mobile robots so that each specific task can be achieved efficiently. More specifically, we study the following four problems: In the *robotic router* problem, our goal is to find minimum number of robots and their strategies to maintain the connectivity of a mobile user to a base station. In the second problem, the *communication bridge* problem, we aim to create a communication bridge between two static end-points by minimizing the number of robots and their movements. In the *data mules* problem, the goal is to find paths for robots so that the sum of the time spent traveling and the time spent to download data from all sensors is minimized. The *multi-robot patrol* problem asks to find the robot patrol paths where a set of locations are visited as frequently as possible according to their priorities.

Next, we introduce these problems and our contributions.

1.2.1 Robotic Routers

In the robotic router problem, our goal is to find the minimum number of robotic routers and their strategies to create an adaptive network so that the connectivity between a stationary base station and a mobile user is maintained at all times. The robotic router problem was first introduced in our work [19]. In a related work [20], Kutylowski et al. present a strategy for using a chain of robots to create a communication chain between a stationary camp and a mobile explorer. Similarly, in [21], the authors propose a chaining formation approach to provide end-to-end communication. They provide solutions to minimize signal-to-noise ratio of the communication links. However, in these works the environments are assumed to be open spaces. In practice, we need to deal with more complex environments than open spaces such as environments with obstacles. Hence, we need topologies more general than communication chains to maintain the connectivity of the user.

In other related work, Stump et al. propose two metrics for characterizing connectivity and present a framework which chooses the best local decision to maintain the connectivity of an independently moving target [22]. However, none of the previous works provides performance guarantees on the number of robots used in their solution. In Chapter 2, we fill this gap by presenting algorithms with provable performance guarantees. Next, we elaborate on these contributions.

Our Contributions

In Chapter 2, we present two solutions for the robotic routers problem. One crucial parameter in a robotic router system is the communication model which tells us whether two nodes are connected or not in their current configuration. Hence, our strategies rely on the communication model assumption.

In Section 2.3, we present a general solution which works for any given communication model. The strategies presented in this solution framework depend on the motion of the mobile user. In the first model, *known-user trajectory*, we assume that the trajectory of the user is known. This is a valid assumption when the mobile user is an entity under our control such as a tele-operated robot. However, in most cases this assumption is not valid. Hence, in the *unknown-trajectory model*, we assume the user's trajectory is unknown and present a strategy which works for any possible user trajectory. We achieve this by finding a strategy which maintains the connectivity of an adversarial user who tries to break the connectivity as quickly as possible. Both strategies presented are optimal. However, the running time of these strategies are exponential in the number of robots. Hence, these solutions do not scale to large systems.

In the second solution (Section 2.4), we overcome the high time complexity by solving a geometric instance of the problem. In this version of the problem, we present algorithms for a user who resides in a geometric environment (e.g. a polygon) and the communication model is defined in terms of geodesic distance. We present an optimal algorithm when the environment is a simply-connected polygon; a constant factor approximation when there is a single obstacle, and an $O(h)$ approximation for geometric environments with multiple obstacles, where h is the number of obstacles.

In Section 2.5, we demonstrate the practical feasibility of robotic routers with simulations and experiments.

1.2.2 Communication Bridge

Given two points on a plane, we use mobile robots to create an on-demand communication between these two end-points. We assume robots are scattered around the environment and an initial communication between the end-points does not exist. In the *communication bridge* problem, our goal is to move the robots so that, in their final locations, there is a communication bridge constructed from robot-to-robot links between the two end-points.

We focus on two measures of efficiency in this problem. The first one is the distance traveled by the mobile robots. We use two metrics for the distance measure: the maximum or the total distance traveled by the robots. When the environment is a graph, Demaine et al. show that minimizing the maximum distance traveled to establish a communication bridge is an NP-hard problem [23]. They present an $O(n)$ solution where n is the number of vertices in the graph. However, this solution is not efficient in terms of the number of robots used when the underlying graph has many vertices. The second measure is the number of robots used in the communication bridge. This measure is important to use minimum number of resources (robots). Moreover, using minimum number of robots reduces the data latency due to hub-to-hub data transfer in the bridge.

Our Contributions

Since the general version of the problem is computationally hard, in Chapter 3 we study a geometric version of the problem where the robots are restricted to move onto the straight line segment joining the two end-points. This assumption is practical since moving the robots to this line segment leads to a bridge with minimum number of robots compared to any curve joining these end-points.

In Section 3.2, we present approximation algorithms for max and sum distance metrics that use minimum of number of robots by deviating from the optimal solution at most $\sqrt{2}$ times in terms of the distance traveled.

1.2.3 Data Mules

We use data mules to autonomously collect data from sparse sensor fields. Most of the work in the literature assumes that the data mule paths are given beforehand [24]. However, computing efficient paths is the most crucial problem as it determines the time to gather data from sensors.

In the data mule problem, we are given the locations of n sensors and aim to find the tours of k robots so that the time to download the data from all sensors is minimized. This formulation appears similar to a well-known NP-hard problem: Traveling Salesperson Problem (TSP). However, there are fundamental differences between our problem and TSP. First, since sensors have a communication range, robots do not have to go all the way to the sensor locations. Rather robots can get close enough to download sensor data. A similar problem arises in TSP with Neighborhoods (TSPN) problem, where n regions are given and the goal is to find the shortest path which visits each region. However, the cost of a TSPN path is measured by its total length. On the other hand, the cost of a data mule path is measured by the sum of time spent for travel and time spent to download data. Note that, since the data mule problem is a generalization of the TSP problem, there exists no polynomial time solution unless $P = NP$.

Our Contributions

For the data mule problem, we have two main contributions. First, we present efficient path-planning algorithms under various communication models. Second, we present a data mule system for monitoring outdoor environments. We demonstrate the practical feasibility of our algorithms through extensive experiments in both indoor and outdoor environments.

In Section 4.3, we present a proof-of-concept indoor data mule system. We present the energy savings achieved by the proposed system by comparing a connected network deployment with our data mule system. In Section 4.4, we further improve the energy savings by improving the link quality between the sensor and the robot. The quality of the link is determined by the packet-loss rate. If the link quality is not good, sensors need to retransmit the data multiple times which leads to unnecessary energy consumption. To overcome this inefficiency, we present a heuristic for the robot to find a good location before starting to download data.

Next, we present path-planning algorithms for data mules. In our previous work [25], we presented an approximation algorithm for collecting data from sensors under disk communication model. In this model, the communication range is a disk centered at the sensor location and the download time inside this disk is constant. However, this model does not address various challenges of radio signals. In the real world, occasionally a robot can hear from sensors even if it is not within the sensor's communication range. In Section 4.5, we present a modification of the previous algorithm to address this real-world challenge. In this modified version, the robot opportunistically downloads from a sensor from which it receives a good signal even though it is not in the communication disk of the sensor.

In order to demonstrate the feasibility and utility of using robots for the data collection task, we develop a low-cost, yet robust, outdoor robot platform. We present the design details of this system in Section 4.5.1. We also demonstrate the performance of our algorithms and the system with extensive field experiments in Section 4.5.2

The disk model ignores the fact that download time can vary depending on the signal strength. Recently, it has been proposed that a *two ring model* better describes the real-world scenarios [26]. In this model, there are two concentric disks centered at the sensor location. Inside the inner disk the communication is reliable thus the expected download time is shorter. Between the boundaries of the inner and the outer disks, communication is possible, however due to increase in packet loss rate, the expected download time increases. In Section 4.6, we present path-planing algorithm for the two-ring communication problem with provable performance guarantees in terms of the total download time.

1.2.4 Multi-Robot Patrol

Multi-Robot Patrol (MRP) problem is the problem of finding strategies for a team of robots in order to visit a given set of locations as frequently as possible [27]. A well-known measure for the performance of an MRP solution is its idleness. The *idleness* of a location is defined as the maximum time interval between two consecutive visits to that location [28]. Although this criterion addresses frequency of visits, it does assume that the importance of each node is identical. On the other hand, in a real application, it might be desirable to patrol some nodes more frequently than others depending on the node's importance [29].

In [30], the authors present a reinforcement learning solution where the reward collected from a node is proportional to the priority and the information gained. However, their solution does not provide any performance guarantee. In this thesis, we present optimal solutions for environments represented by a tree.

Our Contributions

In Section 5, we present a novel version of the multi-robot patrol problem which we call *Weighted Multi-Robot Patrol (WMRP)* problem. In WMRP, the environment is divided into n non-overlapping cells and each cell is assigned a priority (weight) value. The cost incurred by a cell is measured by its *weighted idleness* which is defined as the product of its idleness and weight and the overall cost of a solution is measured by the maximum weighted idleness. Let m be the number of robots, we partition the environment into m non-overlapping regions where each robot is responsible to patrol inside its region.

Our goal is to find a partitioning such that the weighted idleness is minimized. The non-overlapping partitioning is crucial from a practical stand point because the robots can work independently and the collision risk is minimized. We present two optimal solutions. In the first solution (Section 5.3), we present an algorithm for environments whose dual-graph is a tree. In the second solution (Section 5.4), we present an optimal solution which works only with path-like environments but has a better computational complexity than the previous solution.

1.3 Outline

This thesis consists of six chapters including this introductory chapter. Each chapter is self-contained and can be read independently.

In Chapter 2, we present solutions for the general and a geometric version of the robotic routers problem. We also present the practical feasibility of the robotic routers through experiments. A solution to the communication bridge problem is presented in Chapter 3.

Chapter 4 is dedicated to data mule problem. First, we present a proof-of-concept indoor data mule system. Then, we present a heuristic to find a good download location for the robot to start downloading data. Next, we present an opportunistic path-planning approach under disk communication model. We present the design details of an outdoor data mule system and demonstrate the presented algorithm with extensive field experiments. Finally, we present a path-planning algorithm for the two-ring communication model, which better describes the propagation of radio signals.

In Chapter 5, we present a novel version of the Multi-Robot Patrol problem which incorporates the importance of locations to be visited. We present optimal solutions for environments whose dual-graph is a tree or a path.

Chapter 6 concludes the thesis with a discussion of our contributions and directions for future research.

Chapter 2

Robotic Routers

Suppose a mobile user who is operating in a large environment, needs network connectivity to a base station. The user may be a tele-operated robot and the base station may be a gateway to the Internet. Without a communication infrastructure, the mobility of the user would be restricted by the communication radius of the base-station. The traditional solution for providing long-range network connectivity is to deploy a network of static wireless routers which cover the entire area of interest. However when the environment is large, covering it can be costly. Moreover, in some scenarios (such as natural disasters or hazardous conditions) it might be impossible to manually deploy this network in advance.

On the other hand, we can deploy a small number of robots to act as mobile routers. These robots can autonomously relocate themselves according to the movement of the user and maintain the users' connectivity with the base station. In order to demonstrate the potential gain attained by using robots, let us consider the scenario shown in Figure 2.1. In this scenario, a user u navigates inside a semicircular arena, and wishes to remain connected to the base station at c at all times. Suppose the communication range of all devices is σ. If we deploy a stationary network to cover the arena, the number of necessary routers is $\Theta(R^2/\sigma^2)$. Instead, $\Theta(R/\sigma)$ robotic routers can maintain the user's connectivity by staying on the line segment $[cu]$ (the details of a generalization of this strategy are given in Section 2.4.2). Hence, the number of routers used can be drastically reduced by using the mobility of robots.

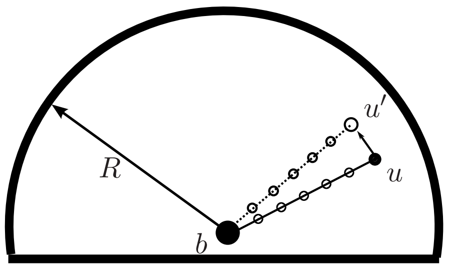


Figure 2.1: An example which demonstrates the potential gain of using robotic routers. We need $\Theta(R^2/\sigma^2)$ static routers whereas $\Theta(R/\sigma)$ robots are sufficient to keep the user connected. When the user moves from u to u', robotic routers move with the same angular velocity to keep the user connected.

Although the previous basic strategy works in a simple environment, more sophisticated strategies are required for more complex environments. For example, when there is an obstacle in the environment (see Figure 2.2-Left), the previous strategy does not work since the obstacles would block the robots' path. When user moves from u to u', the robotic routers have to wrap around the obstacle to maintain the connectivity. However, when the user moves further left (i.e. when user is at u'') the communication constraints are violated and the robots are not able to maintain the user's connectivity. Moreover, adding more robots to this strategy will not help if the user continuously moves around the obstacle. On the other hand, two teams of robots can work cooperatively to maintain the user's connectivity as illustrated in Figure 2.2-Right. In this strategy, a second team of robots (illustrated as square shapes) maintains the connectivity when the user moves further left from u'.

In this chapter, we study the motion strategies of robotic routers to maintain the connectivity of a mobile user to a base station using minimum number of robots. The strategies of the robotic routers rely on the communication model in the environment. In the previous example, a simple communication model of fixed radius σ was used. In a practical application, the communication model in the environment might be more complex due to occlusions (shadowing) and multi-path effects. In Section 2.3, we present robotic router strategies for arbitrary communication models. While this solution is general, its running time is exponential with the number of robots. In Section 2.4, we present geometric solutions where the communication constraint is modeled in terms

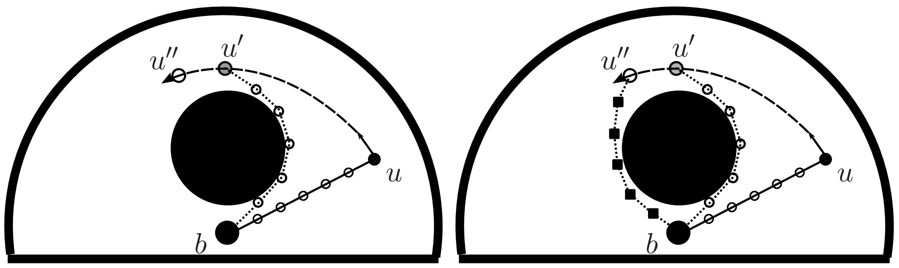


Figure 2.2: When there is an obstacle in the environment, previous simple strategy does not work. Hence better strategies are required for more complex environments. **Left:** If the user moves further left from u'', since the communication constraints are violated, user's connectivity will be broken . **Right:** In a better strategy, a second team of robots (illustrated as square shapes) can maintain the connectivity when robot moves further left from u'.

of geodesic distance. We present polynomial time solutions with provable performance guarantees in terms of the number of robots used in the solution. At the end of chapter, we present simulations and experiments to demonstrate the practical feasibility and advantage of robotic routers.

This chapter is organized as follows: First we present a literature search on robotic routers and related problems in Section 2.1. We present the basic definitions used throughout the chapter in Section 2.2. We present general solutions and geometric solutions for robotics routers in Section 2.3 and Section 2.4, respectively. Finally, we present simulations and experiments in Section 2.5.

2.1 Related Work

A Mobile Ad-hoc Network (MANET) is a network of mobile devices connected through wireless links. MANETs have been extensively studied by the networking community [31]. In a MANET, each mobile device can move independently. Hence, the main challenge is to route the information in a network whose links are constantly changing. Several routing protocols have been proposed in the literature. Optimized Link State Routing (OLSR) [32–34], B.A.T.M.A.N [35], Ad-hoc On-demand Distance Vector (AODV) routing [36], Temporally-Ordered Routing Algorithm (TORA) routing [37,38] and MosquitoNet [39,40] are some of the most known MANET protocols. The main difference of our work from this line of work is due to the controlled mobility. Each device in a MANET is an uncontrolled mobile entity. Whereas in a robotic router network, we use robots' mobility to create an adaptive network between a base station and a target user.

The problem of maintaining connectivity of a team of robots has been studied as rendezvous [41], coalescence [42] and formation [43] problems. In these problems, all network entities whose connectivity needs to be maintained can be controlled. The difference between these problems and the problem addressed in the present work is that, here we are maintaining the connectivity of a target moving independently from the network.

The robotic router problem is also related to some of the control problems in networked multi-robot systems. In the Network Coverage Problem (NCP), the goal is to maximize the coverage or sensing area of a mobile sensing networks [44,45]. In the literature, central [46] and distributed [47–49] approaches are proposed to solve the NCP. Mobile robots are also used for determining gaps in a deployed network and repair it [50–52] as well as improving the connectivity in mobile sensor networks [53,54].

Networked-robots can be also used for performing other tasks while satisfying communication constraints [55]. Distributed flocking algorithms have been proposed for migration of multi-robot networked systems [56, 57]. Communication-aware trajectory [58,59] and target [60] tracking have also been explored. Moreover, a network of robots can be used for improving the communication in search-and-rescue tasks [61,62] and exploration tasks [63].

When the communication model is based on visibility, the Robotic Routers problem is related to the problem of maintaining the visibility of a single moving target. In [64], the authors formulated the problem as a cost minimization problem where they simultaneously penalize the motion of the observer and loss of visibility. In case of known target trajectory, the authors presented a dynamic programming solution which guarantees optimality. For partially-predictable targets, they used a probabilistic motion model for target and presented a solution which maximizes the expected time of visibility. Related work also includes [65] where a sampling-based approach is presented, and [66] where the problem of tracking an evader with a pursuer around a corner is addressed. These problems focus on maintaining the visibility of the target until it disappears for the first time from the field of view of a single robot. In the problems we address, the connectivity model can be more general than visibility and there are additional constraints such as connectivity to the base station. Further, we address the issue of controlling multiple robots.

In other related work, Stump et al. propose two metrics for characterizing connectivity and present a framework which chooses the best local decision to maintain the connectivity of an independently moving target [22]. Here, we are able to give global guarantees by controlling the number of robots. In a related paper [21], Dixon et al. study the problem of forming a chain of robotic relays and present an algorithm to control robots along the chain to improve the signal-to-noise ratio. A similar problem is considered by Kutylowski et al. [20]. They present a global strategy for using a chain of robots to create a communication bridge between a stationary camp and a mobile explorer. In [67], they extend this strategy to local (distributed) strategies. However, in complex environments with obstacles, topologies more general than chains must be used. This is the main focus of the present work. In fact, despite all the recent research effort on this problem, currently there are no algorithms with provable performance guarantees (in terms of the number of robotic routers) to maintain connectivity in complex environments. In this chapter, we present optimal solutions for arbitrary communication models and approximation algorithms for a geometric instance of the problem where communication is determined by geodesic distance.

2.2 Definitions and Notation

Before presenting our algorithms, we start with the basic definitions used throughout the chapter.

A *robotic router* is a mobile robot which can communicate wirelessly. Robotic routers are subject to communication and motion constraints such as limited communication range and a bounded maximum speed. The *base station* is a static entity to which the *user (or target)* wishes to establish connection. All entities are contained in a shared environment denoted by $\mathcal{P}$. The user is connected to the base station through a communicating bridge of robotic routers. We refer to the entities that make up the *robotic router network* (the single user, base station and robotic routers) as *nodes*.

In addition to these entities, we use two concepts frequently: *connectivity* and *motion models*. In the mobile router network, two nodes are *connected* if they satisfy the given connectivity requirements which may depend on the position of nodes, communication range of nodes or possible occlusions. The *user is connected* if it is directly connected to the base station or it is connected through point-to-point links in the mobile router network. The *motion model* of the user is discussed in Section 2.3.1.

2.3 A General Solution

In this section, we provide general solutions to the robotic routers problem for an arbitrary given communication model. Given discretization of the environment, base station location and the number of robots, we present algorithms to create an adaptive network between the base station and a mobile target. We present our algorithms with respect to the motion model of the mobile user. In the first model, we assume that the trajectory of the user is known. This assumption is applicable when the user is a controlled entity, e.g. a tele-operated robot. When the motion model is unknown, we determine our strategies considering a worst-case scenario. In the unknown trajectory model, we assume the user behaves as an adversarial and tries to break the connectivity as quickly as possible. We present a strategy to maintain the connectivity of this adversarial user. Hence, the presented strategy works for all possible user trajectories.

We start with a formulation of the Robotic Routers problem in Section 2.3.1. We present optimal strategies for known and unknown user trajectory in Section 2.3.2 and Section 2.3.3, respectively.

2.3.1 Problem Formulation

In this section, we first present the notation and assumptions specific to the section, and then we formalize the robotic router problem.

Notation and Assumptions

Throughout the chapter, we represent the environment $\mathcal{P}$ as a set of n points. This set contains all possible locations of the nodes (i.e. the user, the base station and the robotic routers). We represent the time domain in unit time steps. All motion models discussed in this section are represented in the discretized domain as a *trajectory*. Let T be the end time of the user motion. The user's trajectory is a sequence u of length T where $u(t)$ is the position of the user at time step t. Similarly, $r_i(t)$ is the position of i^{th} robotic router at time step t. Since the base station does not move, it's trajectory is a constant, b.

A configuration $q = (q_1, ..., q_m)$ is a vector of locations of robotic routers in the mobile router network. As we discretize the time domain, the speed of the user and robotic routers are expressed as step sizes i.e. the distance that a node can move in one time step. In this model, we define the neighbor points that the i^{th} robotic router can move from the point q_j in a single time-step as $N_r(q_j)$ (The subscript r stands for "robot."). To simplify the notation, we assume that all robotic routers have the same speed. One can remove this assumption by defining different neighborhood functions for each robotic router. For the user, we use the neighborhood function N_u. Similarly, $N_c(q)$ is a set of neighbor configurations that can be reached from *configuration* q in one time step. A trajectory r_i is a *valid trajectory* if $\forall t,\ r_i(t+1) \in N_r(r_i(t))$.

Communication Model

We assume that a generic communication model is given beforehand. More specifically, we are given a matrix A such that $A(i, j)$ is 1 if the mobile router network nodes located at i and j can communicate directly and 0 otherwise. Here, to simplify the notation, we assume that the connection range of all nodes are identical for a particular location. The user located on q_u is *connected* by robots in configuration $q = (q_1, ..., q_m)$ if one of the following holds: (i) $A(q_u, b) = 1$ (ii) $\exists q_i$ s.t $A(q_u, q_i) = 1$ and q_i is connected to b through point-to-point link(s) of type (i) or (ii). Let $q(t)$ be the configuration of mobile router network at time step t. The user is *continuously connected* if it is *connected* in $q(t)$ for all $1 \le t \le T$.

Motion Model

In most applications, the environment, location of the base station, wireless range and speed properties of robotic routers do not change significantly. Hence, the trajectory of the user becomes the most important variable in determining robotic router strategies.

In this work, we consider two motion models. In the first model, we assume that we know the trajectory of the user in advance. This assumption is reasonable for some applications, e.g. tele-operated robots control whose trajectory is fixed beforehand. In general, a user may be willing to declare its trajectory when requesting the connectivity service.

However, in some cases it is not feasible to know the user trajectory in advance. In such cases, we may consider the worst case trajectory where user tries to disconnect as quickly as possible. This case analysis can give us a guarantee on whether we can connect the user for any possible trajectory or not. We model this scenario as a *pursuer-evader game* where the user tries to break the connection from the mobile router network as quickly as possible. At the same time, the robotic routers try to extend the connection time for as long as possible, preferably infinitely. We call this user motion strategy as *adversarial user trajectory* and the shortest such trajectory as the *shortest escape trajectory.*

Formulation

In this section, we formalize the robotic routers problem for two motion models: known user trajectory and adversarial user trajectory.

Known user trajectory: Let $\mathcal{P}$ be the environment, A be the connectivity model, b be the position of the base station, m be the number of robotic routers and u be the trajectory of the user. For each robot r_i, find *a valid robotic router trajectory* such that the user is connected to the base station for the maximum possible amount of time.

Adversarial user trajectory: Let $\mathcal{P}$ be the environment, A be the connectivity model, b be the position of the base station, m be the number of robotic routers, q_u be the initial location of the user and q be the initial configuration of mobile router network. Find out whether there exists a user escape trajectory for *pursuer-evader game* where the evader wins by breaking connectivity. If it exists, find the *shortest escape trajectory* u. Compute *valid robotic router trajectories*, that maintain the user's connectivity for as long as possible. Since the user can dynamically change its trajectory during execution, the router strategies for the adversarial case must be adaptive. For this model, we make a design decision and impose the constraint that the robotic router network must remain connected at all times. In other words, each router must remain connected to the base station so that its strategy can be adapted in run time.

In both problems we assume that the number of robotic routers m is given. However, it is easy to obtain the minimum number of routers required for continuous connectivity simply by performing a binary search on m until the continuous connectivity is satisfied. Now, we present strategies for the known user and adversarial user models in Section 2.3.2 and Section 2.3.3, respectively.

2.3.2 Known user trajectory

In this section, we present *KnownUserTrajectory* algorithm for the robotic routers problem when user trajectory is known a priori. The solution uses dynamic programming to obtain robotic routers' trajectories. Recall that $\mathcal{P} = \{x_1, \ldots, x_n\}$ is the environment denoted by a set of points. We build a table C where the entry $C(q, t)$ stores the maximum connection time of the user until time t with routers ending in final configuration $q = (q_1, ..., q_m)$. The table size is $n^m \times T$ where T is the length of the user's trajectory. The first m dimensions of the table correspond to router locations. The i^{th} entry in each of these dimensions correspond to location x_i. Using $C(q, t)$, we find robot trajectories which maximize the connection time of the user. We compute the entry $C(q, t)$ iteratively as follows:

$$C(q,t) = \begin{cases} \max\limits_{q' \in N_c(q)} C(q', t-1) + 1 & \text{if } u(t) \text{ is connected by } q \\ \max\limits_{q' \in N_c(q)} C(q', t-1) & \text{otherwise.} \end{cases}$$

$$C(q,1) = \begin{cases} 1 & \text{if } u(1) \text{ is connected by } q \\ 0 & \text{otherwise.} \end{cases}$$

The value: $\max\limits_{\forall q} C(q, T)$ is the maximum connection time for the given user trajectory u. *KnownUserTrajectory* algorithm can be easily modified to return the corresponding robotic router trajectories $r_1, r_2, ..., r_m$ by backtracking as follows. If $\max\limits_{\forall q} C(q, T) = T$, then there exist robotic router trajectories which keep the user continuously connected. Otherwise, we find robotic router trajectories which maximize the connection time of the user. Let $C(q, T)$ be the entry with maximum value among all entries at the last time step (i.e., $q = \arg\max_\rho C(\rho, T)$). In our output trajectory, q will be the final configuration of robotic routers, i.e. $q(T) = q$. Next, we find the entry $C(q', T-1)$ from the previous time-step such that $q \in N_c(q')$ and $C(q', T-1)$ is maximized [1] . In the output trajectory, q' corresponds to the configuration at time step $T-1$: $q(T-1) = q'$. We continue this computation backwards in time and for each time step, find the robot configuration that maximizes overall connectivity time.

The correctness and optimality of *KnownUserTrajectory* algorithm can be proven directly by induction on t. The running time of the algorithm is determined by the computation time of $C(q, t)$. There are $n^m \times T$ entries. For each entry, computing the maximum value takes $O(n^m)$ time. Hence, the running time is $O(Tn^m)$.

2.3.3 Adversarial user trajectory

In this section, we present *AdversarialUserTrajectory* algorithm for the robotic routers problem for the case where the user moves in such a way that it tries to break the connection as quickly as possible. In contrast to the known trajectory case where the output consists of fixed robot trajectories, the output for the adversarial case is a *strategy* which encodes the appropriate response to every possible user move. The strategy is encoded in Table E where the entry $E[q_u, q]$ corresponds to the positions of the user and robotic routers at an arbitrary time step. Each element of tuple q and q_u is chosen from all possible locations $\mathcal{P}$ of size n. Hence, the dimension of the table is equal to the number of mobile entities in the network $(m + 1)$ and the size of the table is n^{m+1}.

The entries of E are filled using the following algorithm:

In the first step (line 1), we initialize all the entries to infinity. At the end of the algorithm, the entry $E[q_u, q]$ gives the length of the shortest escape trajectory starting from user location q_u and robotic routers' initial configuration q. We will show that if an entry $E[q_u, q]$ remains at infinity at the end of the algorithm, then there exists no escape trajectory. In lines 2-6, we set $E[q_u, q] \leftarrow 0$ if the robotic router network (i.e. either user or one of the robots) is not connected. Here, we constrain the entire robotic router network to be connected rather than requiring only the user's connectivity as in the known trajectory case. The reason why we use the specified connectivity condition

[1] If there are multiple entries which maximize this value, then there are multiple optimal router trajectories. When this happens, we choose an entry that requires the least amount of movement to reach the configuration in the next time-step.

Algorithm 1 AdversarialUserTrajectory

Environment: $\mathcal{P}$, Communication Model: A, Neighborhood Function: N

```
1: $\forall q_u \forall q \;\; E[q_u, q] \leftarrow \infty$
2: for $\forall q_u \forall q$ do
3:   if robotic router network is not connected then
4:     $E[q_u, q] \leftarrow 0$
5:   end if
6: end for
7: for $k = 1$ to $n^{m+1} - 1$ do
8:   for $\forall q_u \forall q$ do
9:     if $\min_{q'_u \in N_u(q_u)} \max_{q' \in N_c(q)} E[q'_u, q'] = k - 1$ then
10:      $E[q_u, q] \leftarrow k$
11:    end if
12:  end for
13: end for
```

is due to the dynamic nature of the adversarial user case. Since the user's trajectory is unknown, its location at time t becomes available to the network only at time t. In our system, the appropriate response is stored at the base station and corresponding motion commands are sent to the robots. Therefore, all robots must remain connected to the base station at all times.

After the initialization steps, we repeat the procedure between lines 8-12 for $n^{m+1}-1$ times. In this procedure, we apply the min-max relation (line 9) to the entire table. Since all configurations where the network is disconnected is already set to 0 in lines 2-6, the maximization is implicitly over configurations where the network remains connected. In each iteration k, we set $E[q_u, q]$ only if the shortest escape trajectory length is k.

With an additional step, we can find if there exists an initial robotic router configuration q corresponding to the initial user location q_u which satisfies the connectivity. If $\exists q,\ E[q_u, q] = \infty$, then we can initialize our robotic routers to configuration q to keep the connectivity uninterrupted. Moreover, if $\forall q_u \exists q,\ E[q_u, q] = \infty$, we can say that m robotic routers are sufficient to maintain the connectivity independent from the initial location and the trajectory of the user.

Next, we explain (i) how we can extract the shortest escape trajectory and the corresponding robotic router trajectories using table E, and (ii) how we can use E to find robotic router trajectories to maintain the connectivity of a user in a system where the trajectory is unknown.

We extract the shortest escape trajectory and the corresponding robotic router trajectories as follows. Let q_u be the initial position of the user: $q_u(1) = q_u$. First, we find the corresponding robotic router positions q which maximize the escape trajectory length, i.e. $q = \arg\max_\rho E[q_u, \rho]$. In the first time step, we will place the robots at locations given by q. Suppose $E[q_u, q] = k$. This means that the length of the escape trajectory is k. We find the next positions of the user and robotic routers by finding the entry $E[q'_u, q'] = k - 1$ where $q'_u \in N_u(q_u)$ and $q' \in N_c(q)$. We update the trajectories according to the new positions: $q_u(2) = q'_u$ and $q(2) = q'$. Similarly, we continue for k steps and extract the rest of the shortest escape trajectory and corresponding robotic router trajectories from E. The correctness and optimality of this algorithm is proven in Theorem 1.

In addition to extracting the shortest escape trajectory, we can use E to find robotic router trajectories when we do not know the user trajectory. We assume that the table E is stored at the base station. Further, we require the user to report its current location when it makes a connectivity request to the network. Afterwards, whenever the user makes its next move, it informs the network about its next location. Note that this happens during run-time. For the new location, the base station looks up the new locations of routers which maximize the connection time of the user. Even when the user moves adversarially, the connectivity is ensured for at least the number of steps given by the entry of E that corresponds to the nodes' current locations. As an example, assume that the user starts at position q_u and robotic routers start with configuration q. The user's connectivity is guaranteed for $E[q_u, q]$ steps. If the user's next location is q'_u then we move robotic routers to the configuration $q' = \arg\max_\rho E[q'_u, \rho]$ such that $q' \in N_c(q)$. This guarantees that the user will be connected for at least $E[q'_u, q']$ time-steps.

The running time of the algorithm is $O(n^{m+1})$: there are n^{m+1} entries and for each iteration we scan the entire table which takes $O(n^{m+1})$ time. We iterate this procedure for n^{m+1} times which leads to the claimed running time.

Theorem 1. *Suppose there exists a shortest escape trajectory such that robotic routers* *are initially in configuration q and user is at location q_u. Let $e(q_u, q)$ be the length of this trajectory.*

1. *$E[q_u, q] = k$ if and only if the length of the shortest escape trajectory $e(q_u, q)$ is k.*
2. *$E[q_u, q]$ is ∞ if and only if there exist a robotic router trajectory which satisfies continuous connectivity for any possible user trajectory.*

Proof. We prove the two statements in order.

Proof of (1):

We show that $E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$ by induction on k.

Basis: $E[q_u, q] = 0 \Leftrightarrow e(q_u, q) = 0$ holds due to the initialization step between lines 2-6. If the escape trajectory is of length 0, this means that the user is disconnected in the initial configuration and we set $E[q_u, q] \leftarrow 0$. Similarly, if $E[q_u, q]$ is set to zero in the initialization step, there exists a trivial escape trajectory of length 0.

Inductive step: let us assume that $\forall k,\ E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$ holds. We show that $E[q_u, q] = k+1 \Leftrightarrow e(q_u, q) = k+1$. We prove this statement by showing that both directions of the conditional statement hold.

First we prove: $E[q_u, q] = k+1 \Rightarrow e(q_u, q) = k+1$. For contradiction, suppose that $E[q_u, q] = k+1$ but $e(q_u, q) \neq k+1$. Due to the inductive step we have: $e(q_u, q) \geq k+1$ (Condition 1). This is because, due to the inductive hypothesis, $e(q_u, q) < k+1$ would imply $E[q_u, q] < k+1$, which is a contradiction. When $E[q_u, q]$ is set to $k+1$, due to the min-max relation, following holds: $\exists q'_u \in N_u(q_u),\ \exists q' \in N_c(q)$ such that $E[q'_u, q'] = k$ and $\forall q'' \in N_c(q),\ E[q'_u, q''] \leq k$. From the inductive hypothesis and the inequality: $\forall q'' \in N_c(q),\ E[q'_u, q''] \leq k$, we have $\forall q'' \in N_c(q),\ e(q'_u, q'') \leq k$. This gives us $e(q_u, q) \leq k+1$ (Condition 2). This is because the user can choose to go to q'_u and follow an escape trajectory of length k afterwards. From conditions (1) and (2), we have $e(q_u, q) = k+1$ which contradicts with the original claim. Thus, $E[q_u, q] = k+1 \Rightarrow e(q_u, q) = k+1$ holds (Condition 3).

Next, we prove: $e(q_u, q) = k+1 \Rightarrow E[q_u, q] = k+1$. Again, for contradiction, let us assume that $e(q_u, q) = k+1$ but $E[q_u, q] \neq k+1$. From the inductive hypothesis, $E[q_u, q] \geq k+1$ holds (Condition 4). Let π be an escape trajectory of length $e(q_u, q) = k+1$ with initial positions of the players given by q_u and q. Let $q'_u \in N_u(q_u)$ be the user location in the second step of π. Since, the escape trajectory length is exactly $k+1$, $\forall q'' \in N_c(q)$, $e(q'_u, q'') \leq k$. Because, otherwise robotic router network can increase the connection time by going to q' where $e(q'_u, q') > k$. Moreover, $\exists q' \in N_c(q)$, such that $e(q'_u, q')$ is exactly k (otherwise by going q'_u, user achieves an escape trajectory of length less than $k+1$ which is a contradiction). By the induction hypothesis: $\forall q'' \in N_c(q)$, $E[q'_u, q''] \leq k$, thus applying the min-max relation yields $E[q_u, q] \leq k+1$ (Condition 5). From conditions (4) and (5), we have $E[q_u, q] = k+1$. This is a contradiction with the original claim. Therefore $e(q_u, q) = k+1 \Rightarrow E[q_u, q] = k+1$ holds (Condition 6).

From conditions (3) and (6), the inductive step is proven. Finally, we showed: $\forall k$ $E[q_u, q] = k \Leftrightarrow e(q_u, q) = k$.

Proof of (2):

The proof of the second statement is straightforward. $E[q_u, q]$ is either marked as $k \leq n^{m+1}$ or ∞ and the user either has a shortest escape trajectory of length $e(q_u, q) \leq n^{m+1}$ or it can not avoid the connection from the robotic router trajectory. Since the number of iterations in the algorithm can not exceed n^{m+1}, the claim above holds for $E[q_u, q]$. Let us assume that there exists an escape trajectory and its length is: $e(q_u, q) > n^{m+1}$. Since the number of permutations of tuples: (q_u, q) is n^{m+1}, we can find a cycle in the sequence of tuples. However, then we can find a shorter escape trajectory by avoiding the cycle which is a contradiction. $\square$

2.4 A Geometric Solution

In the previous section, we presented general solutions for robotic router problem which works for arbitrary communication models. However, the running time of the previous algorithms are exponential with the number of robots. Hence, these solutions might not be scalable for the applications where the number of robots is large. In this section, we present robotic router strategies for geometric environments where the communication model is defined in terms of geometric constraints. We present an optimal algorithm for simply-connected polygons and a constant factor approximation for polygons with single obstacle. Then, we extend these algorithms to polygons with multiple obstacles. The running time of our solutions are polynomial with provable performance guarantees in terms of the number of robots used.

We start with the formulation of the problem in Section 2.4.1. In Section 2.4.2, we present an optimal solution for simply-connected polygons. In Section 2.4.3, we present a constant-factor approximation for a polygonal environment with a single obstacle. In Section 2.4.4, we present an $O(h)$-approximation algorithm for polygons with multiple obstacles. In Section 2.4.5, we present instances which shed further light on the approximation quality of our algorithms.

2.4.1 Problem Formulation

First, we present the terminology and notation used throughout the section, and formalize the robotic router problem.

Notation and Assumptions

Throughout the section, we assume that the time domain is continuous. We denote the position of the user at time t as $u(t)$, and that of the i^{th} robotic router as $r_i(t)$. We assume that both the robotic routers and the user have the same maximum speed. We call this requirement as *the motion constraint*. We will prove the correctness of our strategies by showing that when the robots execute our strategy, the user remains connected to the base station at all times, and the speed of each robotic router at time t never exceeds the speed of the user at time t. In other words, let $|\dot{u}(t)|$ and $|\dot{r_i}(t)|$ be their respective speeds; we present strategies in which $|\dot{r_i}(t)| \leq |\dot{u}(t)|$ always holds.

We measure the distance between any two points $x, y \in \mathcal{P}$ by the length of the geodesic path from x to y, i.e. the shortest path from x to y that lies inside $\mathcal{P}$ and does not cut through any obstacles. For any time t, we denote the geodesic shortest path from the base station b to the user $u(t)$ as $SP(t)$. The shortest geodesic distance between x and y is denoted by $d(x, y)$.

Communication Model

Various models for radio propagation are studied in the literature. Due to various environment dependent effects (such as multi-path, fading, occlusion, etc.), it is difficult to provide a generic model which incorporates all these effects. In this work, we assume that two points $x \in \mathcal{P}$ and $y \in \mathcal{P}$ are connected if $d(x, y) \leq \sigma$ holds. This is the *communication constraint*. This model is empirically justified the next section (i.e. Section 2.5). In addition to fading effects (through the distance threshold), this model implicitly addresses occlusion (shadowing) effects: If there exists a line-of-sight between x and y the geodesic distance is same as the Euclidean distance. However, when the polygon or an obstacle occludes between x and y, the geodesic distance increases.

Any subset of points A in $\mathcal{P}$ is said to be geodesically convex if for every x and y in A, the geodesic segment $[x, y]$ is contained in A. We define the geodesic convex hull of a set of points X in $\mathcal{P}$ as the minimal geodesically convex set that contains all of the points in X.

To simplify the notation, we scale all distances by the communication distance threshold σ. Throughout the paper, without loss of generality, we assume that the *communication distance* is the unit distance. Let D be the longest geodesic shortest path from b to any point $\in \mathcal{P}$; $m^* = \lceil D - 1 \rceil + 1 = \lceil D \rceil$ is a lower bound on the minimum number of robotic routers necessary to connect any point in $\mathcal{P}$ to b, including the base station as a robotic router.

Motion Model

We define the number of robots used as follows. For a given user trajectory $\mu = u(t)$, let $n(\mu)$ be the number of robots required to connect the user to the base station. For a given environment, the number of robots required is the maximum number over all possible user trajectories, i.e. $n = \max_\mu n(\mu)$. When computing n, we do not require that the routers know the user's trajectory in advance. The user can execute any strategy and this information is not available to the robots. However, we require that the robotic routers in the network are all continuously made aware of the current position of the user[2] and they can instantaneously choose their movements based on this information.

Formulation

Given an environment $\mathcal{P}$ (possibly with obstacles) and a base station $b \in \mathcal{P}$, find the minimum number of robotic routers and their motion strategies such that the user u is connected to the base station at all times, and the motion and communication constraints are satisfied.

2.4.2 Environments with no obstacles

In this section, we present a strategy to maintain connectivity using an optimal number of $m^* = \lceil D \rceil$ routers. The strategy, which we call *EQ-DIST*, involves maintaining an equidistant separation along $SP(t)$. We show that this can be achieved without violating communication and motion constraints.

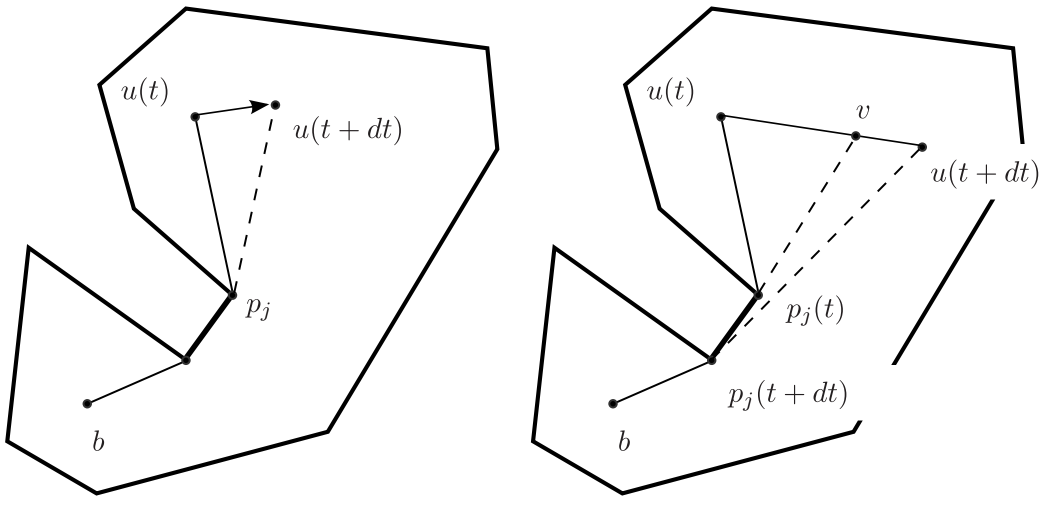
We say that the *Evenly Spaced Property (ESP)* holds at time t if all the routers are positioned uniformly along $SP(t)$. We will refer to this chain of routers as an *arm*. We assume that ESP holds at time 0 (i.e. the user is willing to wait until the initial connection is established).

When the environment $\mathcal{P}$ is a polygon, it is known that $SP(t)$ is a polygonal path $\{p_0 = b, p_1, p_2, \ldots, p_j, u\}$ from b to u, where any p_i for $i > 0$ is a vertex of $\mathcal{P}$. On this path, the *parent* of any point p on SP is defined as the closest vertex of $\mathcal{P}$ to p that lies on the shortest path between b and p (Figure 2.3(a)). If no such vertex exists, then the parent of p is b.

First, we prove the following lemma.

Lemma 2. *For any t there exists a sufficiently small $dt > 0$ and a shared point $s \in SP(t) \cap SP(t+dt)$ such that $SP(t)$ and $SP(t+dt)$ only differ along a single line segment, from their respective endpoints to s.*

[2] For example, this information can be provided by the user.



(a) Parent does not change (b) Parent changes

Figure 2.3: For a small enough time step, the shortest path $SP(t)$ changes only along the line segment farthest from b.

Proof. First, observe that for all t and for all dt, $SP(t)$ and $SP(t+dt)$ share their start point b, by definition. Therefore, s always exists. We now show that we can find a dt and a shared point s such that the two shortest paths differ only along their last line segment.

Take any time interval $dt > 0$. The user moves from $u(t)$ to $u(t+dt)$. Let $p_j(t) \in SP(t)$ be the parent of $u(t)$, and $p_j(t+dt) \in SP(t+dt)$ be the parent of $u(t+dt)$. There are two possibilities, as shown in Figure 2.3.

Case 1. (Figure 2.3(a)) $p_j(t) = p_j(t+dt)$, i.e. the parent of the user remained the same over the interval dt. Then $s = p_j(t) = p_j(t+dt)$ is the shared point.

Case 2. (Figure 2.3(b)) $p_j(t) \neq p_j(t+dt)$, i.e. the parent of the user changed over the interval dt. The parent for any point on a shortest path to b changes only when the new shortest path wraps (or unwraps) around a vertex of $\mathcal{P}$. Here, we discuss the case when it unwraps, i.e. $p_j(t+dt)$ is an ancestor of $p_j(t)$. The case when it wraps is similar.

Let $p_{j-1}(t)$ be the parent of $p_j(t)$. Shoot ray L along line segment $[p_{j-1}(t)\ p_j(t)]$. There are two possibilities. If L does not intersect $[u(t)\ u(t+dt)]$, then the parent of the user did not change, which contradicts our assumption in Case 2. Let v be the point

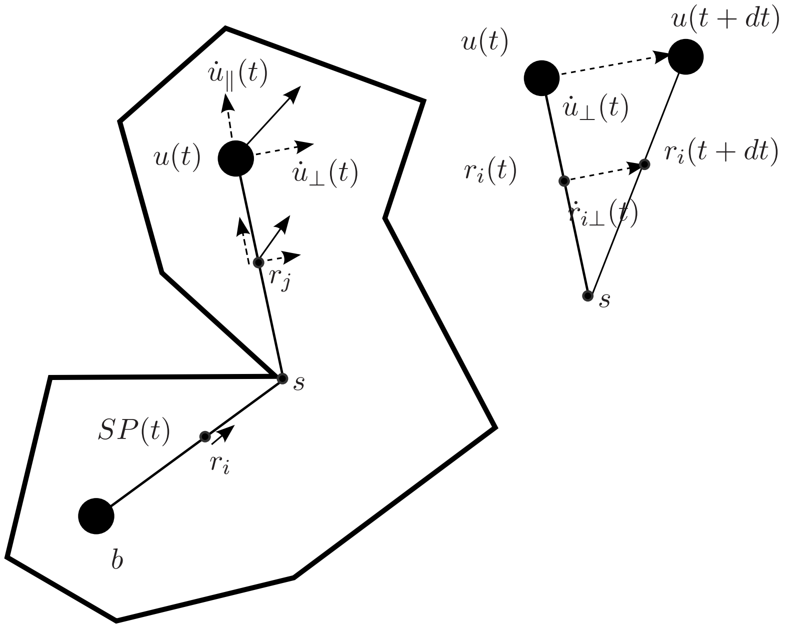


Figure 2.4: **Left:** Robots r_i only translate along SP while robots r_j rotate about s and translate. **Right:** Similar triangles used to find the relation between $|\dot{u}_{\perp}(t)|$ and $|\dot{r}_{i\perp}(t)|$.

of intersection of L and $[u(t)\ u(t+dt)]$. Since the user follows a continuous path from $u(t)$ through v to $u(t+dt)$, there exists $0 < dt' < dt$ for which $u(t+dt') = v$. Now $s = p_j(t)$ is the desired shared point common to $SP(t)$ and $SP(t+dt')$. Hence, $SP(t)$ and $SP(t+dt)$ only differ in their last line segment when $dt = dt'$. □

We now show that the robots can maintain connectivity using *EQ-DIST*:

For any point $z(t) \in SP(t)$, let $z^{\dagger}(t)$ denote its parent. Parameterize the velocity $\dot{z}(t)$ into a radial component $\dot{z}_{\|}(t)$, along $[z^{\dagger}(t)\ z(t)]$, and a tangential component $\dot{z}_{\perp}$ orthogonal to $\dot{z}_{\|}(t)$. We have $|\dot{z}(t)| = \sqrt{|\dot{z}_{\perp}(t)|^2 + |\dot{z}_{\|}(t)|^2}$. For any robot $r_i(t)$, we denote its velocity components as $\dot{r}_{i\|}(t)$ and $\dot{r}_{i\perp}(t)$ (see Figure 2.4).

Only the radial component of the user's velocity affects the length of SP. Let λ be a differential change in the length of SP. We have $\lambda = \dot{u}_{\|}(t)$. To satisfy ESP, the robotic routers should move proportional to λ along the radial component.

$$\dot{r}_{i\|}(t) = \frac{i}{n+1}\dot{u}_{\|}(t) \tag{2.1}$$

When the user moves, some line segments along SP rotate, while others remain the same. The tangential velocity of any robot is thus a function of which side of the shared point s the robot lies on.

If r_i lies between s and u, we can show using similar triangles (see Figure 2.4) that

$$\frac{|\dot{r}_{i\perp}(t)|}{|\dot{u}_{\perp}(t)|} = \frac{||s\ r_i(t)||}{||s\ u(t)||}$$

Where $||s\ r_i(t)||$ is the length of line segment $[s\ r_i(t)]$ and $||s\ u(t)||$ is the length of line segment $[s\ u(t)]$. Since $r_i(t)$ is closer to b than $u(t)$, we have $||s\ r_i(t)|| \leq ||s\ u(t)||$, i.e.

$$|\dot{r}_{i\perp}(t)| \leq |\dot{u}_{\perp}(t)| \tag{2.2}$$

The robots between s and b have a tangential component of zero.

Therefore, for any robot, (2.1) and (2.2) show that the robot only needs to move at most as fast as the user to stay on the geodesic from b to u while maintaining ESP.

Theorem 3. *In a simply-connected polygon, the number of mobile robots that the EQ-DIST strategy requires is optimal.*

Proof. Recall that the cost of the optimal solution is the required number of robots to connect any user trajectory. When the user goes to a location where SP is maximized, the optimum solution has to use at most $m = \lceil D \rceil$ robots. We have seen that *EQ-DIST* can maintain connectivity using the same number of robots. $\square$

2.4.3 Environments containing a single obstacle

In this section, we present robotic router strategies where a user is connected to b in a polygonal environment with a single obstacle. First we present a solution for circular obstacles, then we show how to extend this solution to convex and non-convex obstacles.

Circular obstacle

Let $\mathcal{O}$ be a circular obstacle and let c and r be the center and radius of $\mathcal{O}$. Our strategy is as follows. First, we connect every point on $\mathcal{O}$ to b by extending an arm starting from b and wrapping it around $\mathcal{O}$. We call this our *wrapping arm.* The robotic routers in it will remain stationary (see Figure 2.5). After connecting $\mathcal{O}$ to b, we use a *connecting arm* which rotates around $\mathcal{O}$ and connects the user to $\mathcal{O}$ which is then connected to b through the wrapping arm.

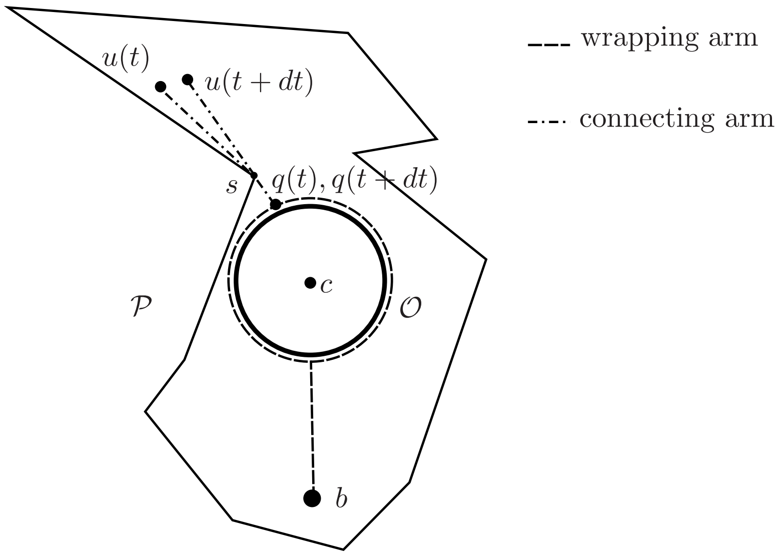


Figure 2.5: Illustration of the first case of our strategy.

We place robots so that ESP is satisfied; these locations can be easily found by using geometric properties of lines and circles. The bounds on the length of the arm and the number of robots used will be obtained in Theorem 6.

The connecting arm's responsibility is to connect user to $\mathcal{O}$ and consequently to b. We achieve this by moving robotic routers on the Shortest Path (SP) between u and $\mathcal{O}$. To guarantee that the connecting arm is always connected to $\mathcal{O}$, we use an additional robotic router q which moves along the boundary of $\mathcal{O}$. Robot q acts as a base station for the connecting arm. Let $SP_{\mathcal{O}}(t)$ be the shortest geodesic path between $\mathcal{O}$ and $u(t)$ (this path is the subset of SP between c and $u(t)$). Robot q always remains at the beginning of this path on $\mathcal{O}$. Recall that the parent of $u(t)$ is the closest vertex to $u(t)$

in $SP_{\mathcal{O}}(t)$ (We treat q as a vertex as well.).

We analyze the connecting arm strategy in two cases: (i) $u(t)$ has a parent different than $q(t)$ (ii) $u(t)$ has $q(t)$ as the parent.

Case (i): If there exists a parent s of $u(t)$ such that $s \neq q$, then we can find a dt and a shared point s such that the shortest paths $SP_{\mathcal{O}}(t)$ and $SP_{\mathcal{O}}(t+dt)$ differ only along their last line segment (Lemma 2). Since both shortest paths pass through s and the shortest path from $\mathcal{O}$ to s is same, q does not move, i.e. $\dot{q}(t) = 0$ (See Figure 2.5). In this case, the connecting arm can execute EQ-DIST and maintain connectivity.

Case (ii): If the parent of $u(t)$ is q, we can move q and the robots on the connecting arm in such a way that they maintain ESP without violating motion and communication constraints. As we did in Section 2.4.2, we divide the velocity $\dot{u}(t)$ into two components: radial velocity $\dot{u}_{\|}$ and tangential velocity $\dot{u}_{\perp}$. Since q is moving on the boundary of $\mathcal{O}$ its radial velocity is 0. If q is the common parent for $u(t)$ and $u(t+dt)$, these shortest paths rotate around c and rotation is due to the tangential component of u (see Figure 2.6). As angular velocity is the same for the user and each robot on $SP_{\mathcal{O}}$, we can conclude that $|\dot{q}_{\perp}(t)| \leq |\dot{r}_{i\perp}(t)| \leq |\dot{u}_{\perp}(t)|$. If $u(t)$ and $u(t+dt)$ do not have q as their common parent, we can show that a time interval $dt' < dt$ can be found such that the above condition holds. The proof is the same as in Lemma 2.

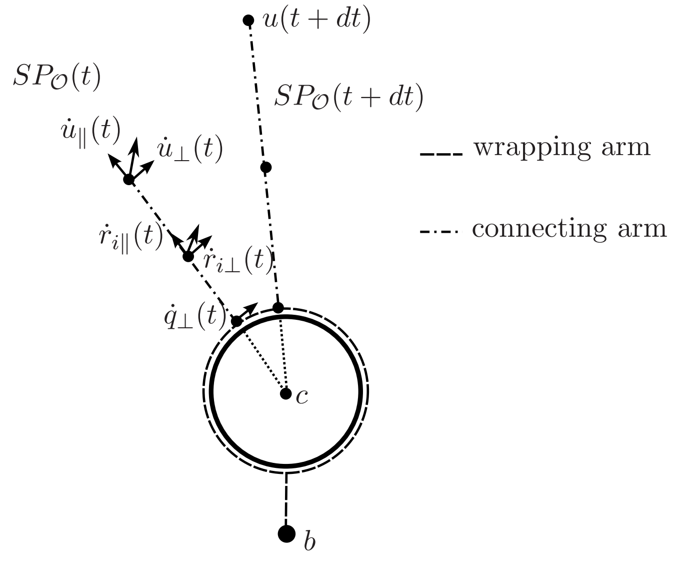


Figure 2.6: Illustration of the second case of our strategy.

Suppose $\dot{u}_{||}(t)$ is positive; in this case $SP_{\mathcal{O}}$ increases in length. To satisfy ESP, the robotic routers have to move towards u. The distance from q to r_i must increase by $\frac{i}{n_c}|\dot{u}_{||}(t)|$ where n_c is the number of robots in the connecting arm, including q, and i is the robot index in the connecting arm (the 0^{th} robot refers to q). Hence, $0 = |\dot{q}_{||}(t)| \le |\dot{r}_{i||}(t)| \le |\dot{u}_{||}(t)|$ holds. Together with the constant angular velocity observation, we conclude that $|\dot{q}(t)| \le |\dot{r}_i(t)| \le |\dot{u}(t)|$ and the motion constraint is satisfied.

We now bound the number of robots used by this strategy.

Lemma 4. *The length of the connecting arm is upper bounded by* $2D$*.*

Proof. Let p_c and p_f be the closest and furthest points on $\mathcal{O}$ from b, respectively. By definition, we know that $d(b,p_c) \le d(b,p_f) \le D$ where D is the maximum SP from b.

We find an upper bound on the length of the connecting arm using triangle inequality. For any point x in the polygon, due to triangle inequality, we have $d(c,x) \le d(c,b) + d(b,x)$. We subtract r from both sides: $(d(c,x) - r) \le d(x,b) + (d(b,c) - r)$. The connecting arm has length $d(q,x)$ where q is the closest point on $\mathcal{O}$ from x. This distance is equal to the left hand side of the inequality, i.e. $d(c,x) - r$. By definition $d(x,b)$ and $d(b,p_c) = d(b,c) - r$ are upper bounded by D, hence the length of the connecting arm is upper bounded by $2D$ (Figure 2.7).

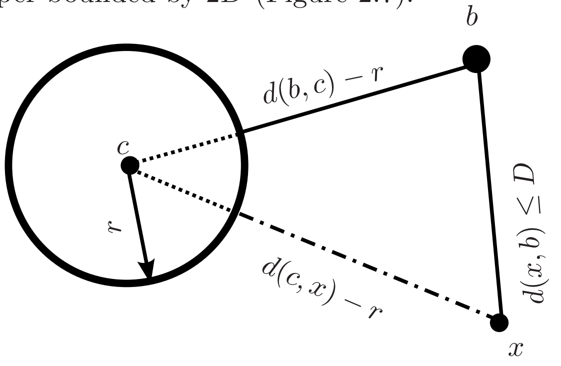


Figure 2.7: Bounding the length of the connecting arm using triangle inequality.

□

Lemma 5. *The length of the wrapping arm is upper bounded by* $3D$*.*

Proof. Next, we find a bound on the length of the wrapping arm (Figure 2.8). The length of the wrapping arm is equal to the sum of the SP distance from b to $\mathcal{O}$ and the circumference of $\mathcal{O}$, i.e. $2\pi r + d(b, p_c)$. Since $d(b, p_c)$, the length of wrapping arm is upper bounded by $2\pi r + D$. Now we will find a bound on $2\pi r$ by finding a bound on the ratio: $\frac{\pi r}{D}$. Although we do not know D, we use the lower bound on $d(b, p_f)$ to find an upper bound on this ratio. Hence we will show the following inequality holds: $\frac{\pi r}{D} \leq \frac{\pi r}{d(b,p_f)} \leq 1$. First, we calculate the maximum value of this ratio for a special case where $\mathcal{P}$ does not intersect with the tangents $[b\ p_t]$ and $[b\ p_t']$. In this case, the ratio is $\frac{\pi r}{r(\theta + \cot(\theta) + \pi/2)}$. Using basic calculus, we can show that the maximum value of the ratio is 1.

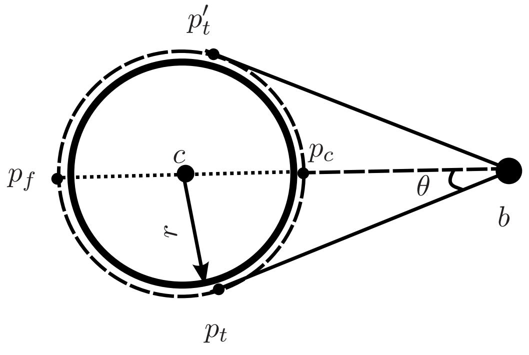


Figure 2.8: The shortest geodesic distance from b to p_f and the length of wrapping arm in a convex polygon.

We now show that if $\mathcal{P}$ intersects with one or both of the tangents, the ratio is reduced. Hence, the upper bound found in the special case is valid for any case.

Two types of vertices of $\mathcal{P}$ exist which can intersect with one or both of the tangents. Figure 2.9 shows these cases. The furthest point p_f on obstacle has the property that it has two shortest paths from the opposite sides of the obstacle. Any other point on the obstacle has a unique shortest path. For example, in Figure 2.8, these shortest paths are $SP_1 = \{b, p_t, p_f\}$ and $SP_2 = \{b, p_t', p_f\}$. Since both shortest paths are equal, we can find $d(b, p_f) = SP_1 = SP_2$. Moreover, since both shortest paths start from b and end at p_f, their union yields a geodesic convex hull around $\mathcal{O}$ and b. Hence, we can say that $d(b, p_f)$ is half of $perim(\mathcal{H})$ where $perim(\mathcal{H})$ is the perimeter of the convex hull $\mathcal{H} = \{b, p_t, p_f, p'_t\}$.

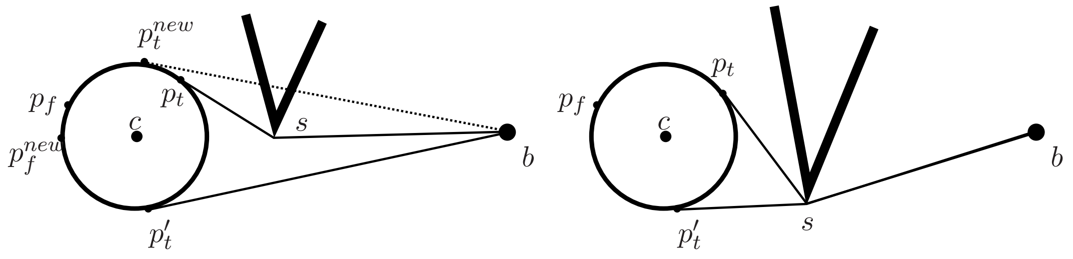


Figure 2.9: Two types of vertices of $\mathcal{P}$ which can obscure with $[b\ p_t]$ and $[b\ p'_t]$

First, we consider the first type of vertex (Figure 2.9-Left). Let s be a vertex which interferes with $[b\ p_t]$. The distance of the furthest point on $\mathcal{O}$ from b is half of $perim(\mathcal{H})$ where $\mathcal{H} = \{b, p_t, p_f, p'_t\}$. Now assume that we remove s; the distance of furthest point on $\mathcal{O}$ from b becomes half of $perim(\mathcal{H}^{new})$ where $\mathcal{H}^{new} = \{b, p_t^{new}, p_f^{new}, p'_t, b\}$. By triangle inequality $(d(b, p_t^{new}) \leq d(b, s) + d(s, p_t) + d(p_t, p_t^{new}))$, we can show that $perim(\mathcal{H})$ is longer than $perim(\mathcal{H}^{new})$. Hence, by introducing s, we reduce the ratio: $\frac{\pi r}{d(b,p_f)}$.

Let s be a vertex of the second type (Figure 2.9-Right) which interferes with both $[b\ p_t]$ and $[b\ p'_t]$. Observe that $\frac{\pi r}{d(s,p_f)} \leq 1$ holds due to the special condition that we discussed before. Because $d(b, p_f) = d(b, s) + d(s, p_f)$, the following inequalities hold: $\frac{\pi r}{d(b,p_f)} \leq \frac{\pi r}{d(s,p_f)} \leq 1$.

If there are multiple vertices of $\mathcal{P}$ interfering with $\mathcal{H}$, this ratio still holds because introducing additional vertices between s and b only increases $d(b, p_f)$. Finally, $\pi r \leq D$ holds and consequently the length of the wrapping arm is upper bounded by $3D$, i.e. $2\pi r + D \leq 3D$. □

Theorem 6. *In a polygon with a single circular obstacle, let m^* be the minimum number of robots required to maintain connectivity. The strategy presented in this section uses at most $5m^*$ robots.*

Proof. From Lemma 4 and Lemma 5 the number of robots used by connecting and wrapping arm are upper bounded by $\lceil 2D \rceil$ and $\lceil 3D \rceil$, respectively. Any solution has to use at least $\lceil D \rceil$ number of robots using the same reasoning in Theorem 3. Hence the number of robots used in our strategy is bounded by 5 times the number of robots used in the optimal solution which completes the proof.

$$\lceil d(b, p_c) + D \rceil + \lceil d(b, p_c) + 2\pi r \rceil \leq \lceil 2D \rceil + \lceil 3D \rceil \leq 5 \lceil D \rceil \leq 5m^*$$

□

Convex polygonal obstacle

In this section, we extend our strategy for single circular obstacles to single convex polygonal obstacles. The main idea of the strategy remains the same. The wrapping arm stretches from b to convex obstacle $\mathcal{O}$ and wraps around $\mathcal{O}$. The connecting arm stays on the shortest path (SP) from b to $\mathcal{O}$ and an extra robot q moves on the boundary of $\mathcal{O}$, bridging the connecting and wrapping arms.

The strategy of the robots on the wrapping arm is straightforward: We place robots so that ESP holds. These robots do not move.

Let k be the number of vertices of $\mathcal{O}$. Let $V = \{v_0, v_1 \ldots, v_{k-1}\}$ and $E = \{e_0, e_1 \ldots, e_{k-1}\}$ be the vertex and edge sets of $\mathcal{O}$, respectively. We split $\mathcal{P}$ into $2k$ regions by shooting two rays from each vertex v_i: one perpendicular to e_i and one perpendicular to $e_{mod(i+1,k)}$ (See Figure 2.10).

There are two types of regions in this partitioning. The regions labeled as $\mathcal{V}_i$ has the property that the shortest path from any point $x \in \mathcal{V}_i$ to $\mathcal{O}$ starts at v_i. Similarly, $\mathcal{E}_i$ has the property that the shortest path from any point $x \in \mathcal{V}_i$ to $\mathcal{O}$ starts at some interior point of e_i (See Figure 2.10).

Let $SP_{\mathcal{O}}(t)$ be the shortest geodesic shortest path between $u(t)$ and $\mathcal{O}$ and parent of $u(t)$ be the closest vertex to $u(t)$ in $SP_{\mathcal{O}}(t)$. We investigate the connecting arm's strategy in three cases: (i) $u(t)$ has a parent which is different than $q(t)$ (ii) $u(t)$ has $q(t)$ as its parent and $q(t)$ is on one of the vertices of $\mathcal{O}$ (iii) $u(t)$ has $q(t)$ as its parent and $q(t)$ is on one of the edges of $\mathcal{O}$.

Case (i) If $u(t)$ has a parent other than $q(t)$ then as shown in Section 2.4.3, we can show that $\dot{q}(t) = 0$ (see bottom strategy in Figure 2.11). The proof is identical to the first case in Section 2.4.3. Finally, we use EQ-DIST strategy to keep the user connected when this condition holds.

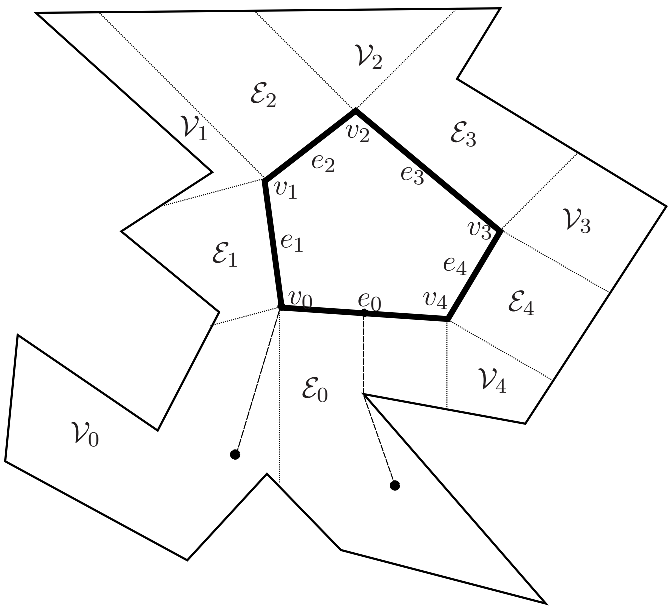


Figure 2.10: The partitioning for single convex obstacle case.

Case (ii) If $q(t)$ is the parent for SP from $u(t)$ to $\mathcal{O}$ and $q(t) \in \mathcal{V}$, we can think of $q(t)$ as a base station (i.e. $\dot{q}(t) = 0$) and execute EQ-DIST strategy as long as this condition holds (see right strategy in Figure 2.11).

Case (iii) When $q(t)$ is the parent of $u(t)$ and $q(t)$ is on an edge e_j, then SP becomes a perpendicular line segment to e_j. Let $SP_{\mathcal{O}}(t)$ and $SP_{\mathcal{O}}(t + dt)$ be the shortest paths at time t and $t + dt$. We divide the velocity of u into two components: radial velocity $\dot{u}_{\|}(t)$ and tangential velocity $\dot{u}_{\perp}(t)$. We make the same division for any robot r_i on the connecting arm including q. Since both shortest paths are perpendicular to e_j, we can show that tangential velocities are equal: i.e. $\dot{q}_{\perp}(t) = \dot{r}_{i\perp}(t) = \dot{u}_{\perp}(t)$ (see Figure 2.12). Since q stays on the boundary, we have $|\dot{q}_{\|}(t)| = 0$. Moreover, we can show that $|\dot{r}_{i\|}(t)| = \frac{i}{n_c}|\dot{u}_{\|}(t)|$ where n_c is the number of robots in the connecting arm and i is the robot index in the connecting arm. Hence, $|\dot{q}_{\|}(t)| \leq |\dot{r}_{i\|}(t)| \leq |\dot{u}_{\|}(t)|$ holds which implies $|\dot{q}(t)| \leq |\dot{r}_i(t)| \leq |\dot{u}(t)|$.

In the above analysis, we assume that in $SP_{\mathcal{O}}(t)$ and $SP_{\mathcal{O}}(t + dt)$ the parent of u is q. If this is not the case, since q is initially the parent, we can find a time interval $dt' < dt$ where q is the parent of u in $SP_{\mathcal{O}}(t + dt')$. After the parent changes, we use

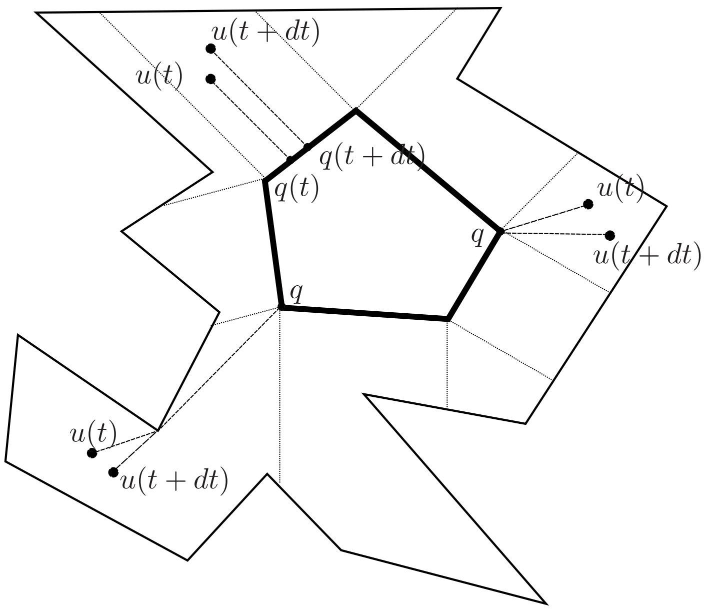


Figure 2.11: The strategies for convex polygon case.

EQ-DIST strategy.

Theorem 7. *In an arbitrary polygon with a single convex obstacle, let m^* be the number of robots used by optimal solution. Our strategy uses at most $5m^*$ robots.*

Proof. The proof is very similar to the proof of Theorem 6. Recall that by using the triangle inequality, we can show that the length of the connecting arm is less than or equal to $d(b,q)+D$ where D is the length of the maximum SP from b and q be the closest point on $\mathcal{O}$ from u. By definition, we have $d(b,q) \leq D$. Hence connecting arm is upper bounded by $2D$. The length of the wrapping arm is equal to $d(b,p_c) + perim(\mathcal{O})$. By definition, we have $d(b,p_c) \leq D$. In the rest of the proof, we show that $perim(\mathcal{O}) \leq 2D$ holds.

Let $\mathcal{H}_b = \{b, v_i, v_{i+1} \ldots, v_k, v_0\}$ be the convex hull of b and $\mathcal{O}$. The distance between b and its furthest point on $\mathcal{O}$ (i.e. $d(b,p_f)$) is equal to the half of the perimeter of the convex hull (i.e. $perim(\mathcal{H}_b)$). Hence, if we show that $perim(\mathcal{O}) \leq perim(\mathcal{H}_b) = 2d(b,p_f)$ then we have $perim(\mathcal{O}) \leq 2D$.

We define the above inequality in terms of edge lengths, i.e. $perim(\mathcal{O}) = e_1 + \cdots + e_i + e_{i+1} + \cdots + e_k + e_0 \leq x + y + e_{i+1} + \cdots + e_k + e_0 = perim(\mathcal{H}_b)$. As shown in Figure 2.13,

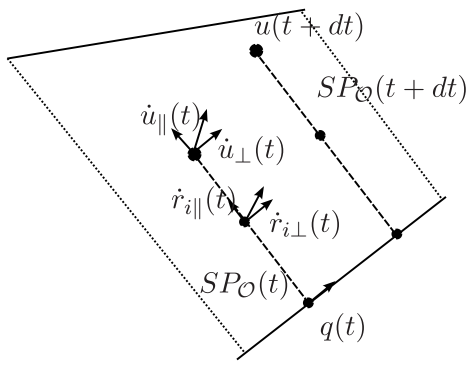


Figure 2.12: We show that in the third case of the strategy ESP and the motion and communication constraints are satisfied.

$\mathcal{H}_b$ and $\mathcal{O}$ has common edges and we can subtract these edges from both sides of the inequality, i.e. $e_1 + \cdots + e_i \leq x + y$. We show that this inequality holds by induction on the number of edges e_i. The base case where $\mathcal{O}$ differs from $\mathcal{H}_b$ with a single edge is trivial (directly from triangle inequality). Let us assume that this inequality holds for all $e_i, i \leq k$, then (i) $e_1 + \cdots + e_i \leq x + y$ holds. Using triangle inequality in the triangle $\triangle bb'v_{k+1}$, we have (ii) $e_{k+1} + y \leq (x' - x) + y'$. Combining (i) and (ii), we get $e_1 + \cdots + e_k + (e_{k+1} + y) \leq x + y + (x' - x) + y'$ which proves the inductive step.

Hence, we showed that $perim(\mathcal{O}) \leq perim(\mathcal{H}) \leq 2D$. Note that $\mathcal{P}$ can occlude $\mathcal{H}$, however we showed in the proof of Theorem 6 that this only increases the perimeter of $\mathcal{H}$ and the same argument holds for this case too. Hence, we skip the proof. Since the above inequality holds, the length of the wrapping arm is upper bounded by $3D$.

The number of robots used by connecting and wrapping arm are upper bounded by $\lceil 2D \rceil$ and $\lceil 3D \rceil$, respectively. Using the $\lceil D \rceil$ lower bound on the number of robots used by the optimal solution, we can upper bound the number of robots used by our strategy as follows:

$$\lceil d(b,q) + D \rceil + \lceil d(b,p_c) + perim(\mathcal{O}) \rceil \leq \lceil 2D \rceil + \lceil 3D \rceil \leq 5m^*$$

$\square$

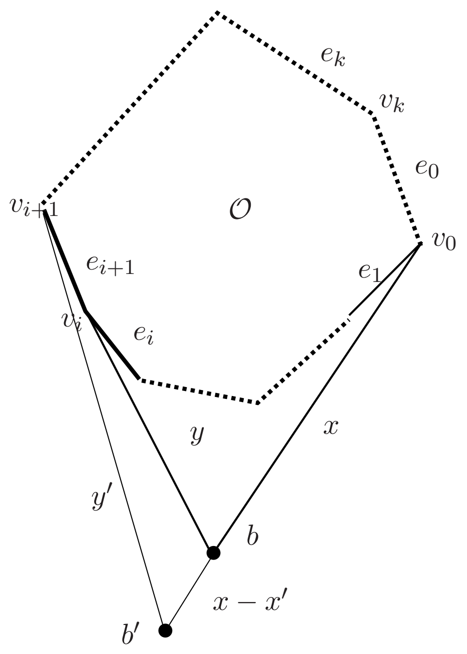


Figure 2.13: We can show that $perim(\mathcal{O}) \leq perim(\mathcal{H})$

Non-convex obstacles

Let $\mathcal{O}$ be a general, possibly non-convex, obstacle. We can extend the strategy for convex obstacles to $\mathcal{O}$ as follows.

Let $\mathcal{H}$ be the geodesic convex hull of $\mathcal{O}$. When u is outside of $\mathcal{H}$, our strategy to maintain connectivity between u and b is similar to what it was when we were dealing with a convex obstacle, but now we treat $\mathcal{H}$ as the obstacle: The wrapping arm stretches from b to $\mathcal{H}$ and wraps around $\mathcal{H}$. The connecting arm stays on the SP from $\mathcal{H}$ to u. We add an extra robot q on the boundary of $\mathcal{H}$ to bridge the connecting and wrapping arms (see Figure 2.14).

When u is inside $\mathcal{H}$, the boundaries of $\mathcal{H}$ and $\mathcal{O}$ create a set of disjoint simply connected regions, and we consider the one that contains u. This region is open only from one side (the boundary of $\mathcal{H}$), and an it does not interfere with any edges of $\mathcal{P}$. Hence, when u is inside $\mathcal{H}$, we can use the same strategy as when it is outside, and move our connecting arm so that it is on the SP between u and the boundary of $\mathcal{H}$.

When u is on the boundary of $\mathcal{H}$, q and other robots in the connecting arm are located at the same location. Hence, when the user moves from the inside region to the outside region (or vice versa) the transition point satisfies both the inside and outside strategies.

In both cases, we are connecting u to the closest point on the perimeter of a convex region with a robot q as a bridge, and we know from previously that this is possible while satisfying the motion constraint.

Theorem 8. *In an arbitrary polygon containing a single polygonal obstacle, let m^* be the number of robots used by the optimal strategy. Our strategy uses at most $5m^*$ robots.*

Proof. The proof is very similar to the proof of the case where the obstacle is convex, except now we use geodesic shortest paths instead of straight lines. Using the triangle inequality as before we can say that the length of the connecting arm is less than or equal to $2D$, where D is the length of the longest $SP \in \mathcal{P}$, and as before, the part of the wrapping arm from b to the closest point in $\mathcal{H}$ is shorter than D.

We define $\mathcal{H}_2$ as the geodesic convex hull of $\mathcal{H}$ and b. We know that $perim(\mathcal{H}_b)$ is $2D$, and we want to prove that $perim(\mathcal{H})$ is not more than that. As before, $perim(\mathcal{H})$ and $perim(\mathcal{H}_b)$ have common edges, and if we subtract these edges from the union of $perim(\mathcal{H})$ and $perim(\mathcal{H}_b)$ we end up with a triangle of shortest paths that connect b and the vertices v_0 and v_i where $\mathcal{H}_b$ first touches $\mathcal{H}$. By applying the triangle inequality to this $\triangle bv_0v_i$ we can conclude that $d(v_0, v_i) \leq d(b, v_0) + d(b, v_i)$, and it follows that $perim(\mathcal{H}) \leq perim(\mathcal{H}_b) \leq 2D$.

Therefore the upper bound on the total number of robots required by our strategy is:

$$\lceil D \rceil + \lceil perim(\mathcal{H}) \rceil + \lceil 2D \rceil \leq \lceil 5D \rceil \leq 5m^*$$

□

2.4.4 Polygonal environments with multiple obstacles

Let $\mathcal{P}$ be a convex polygonal environment containing two or more non-intersecting obstacles. If the convex hulls of the obstacles are disjoint, we can often extend the

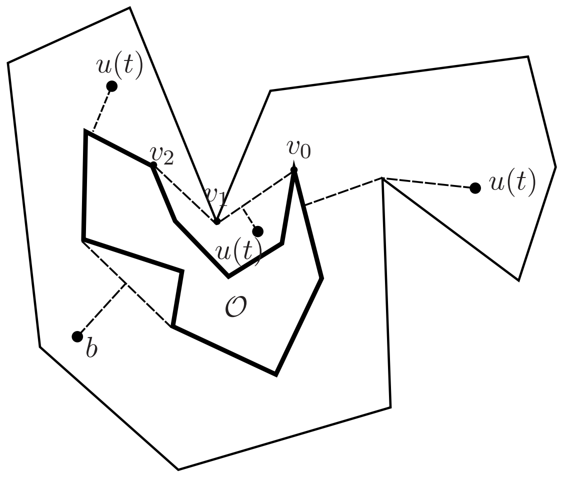


Figure 2.14: An example of our strategy for dealing with a non-convex obstacle

strategies for the single obstacle case as follows. First, we partition $\mathcal{P}$ into convex cells such that each cell contains exactly one obstacle.

For each cell we execute a strategy similar to the one we used in domains with a single obstacle: we have a wrapping arm that connects to b and wraps around the obstacle in the cell, and we have a connecting arm that, whenever u is in the cell, connects u to the closest point to u on the obstacle.

We start by presenting the partitioning strategy for the case when all of the obstacles are circular.

Power diagrams for circular obstacles

Our partitioning strategy relies on the concept of power diagrams [68]. The power $pow(x, s)$ of a point x with respect to a circle (or in our case a circular obstacle) s in the Euclidean space $\mathbb{R}^2$ is given by $d^2(x, z) - r^2$, where d is the Euclidean distance function, and z and r are the center and the radius of s. For a finite set of circles S in $\mathbb{R}^2$, the *power diagram* of S, denoted $PD(S)$, is a cell complex that associates each $s \in S$ with the convex domain $\{x \in \mathbb{R}^2 | pow(x, s) < pow(x, t), \forall t \in S - s\}$. An example is shown in Figure 2.15. When $r = 0$, i.e. the circles degenerate to points, $PD(S)$ becomes the Voronoi diagram. The following properties about $PD(S)$ are relevant to our partitioning strategy (see [68] - §2.2: Observations 1 and 2, and Lemma 1).

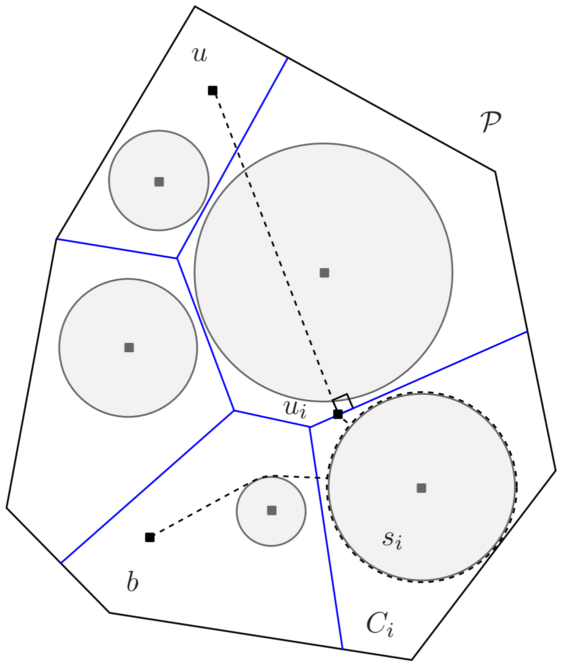


Figure 2.15: Power diagram edges are shown in blue (light color). Circular obstacles are shaded. The connection from b to cell C_i is shown, along with the construction of the point u_i.

- When the circles are non-intersecting, the edges of $PD(S)$ do not intersect any of the circles.
- If the cardinality of S is k, then $PD(S)$ contains at most k cells.

Multiple circular obstacles

Let S be the set of finite circular obstacles in our environment. We intersect each cell in $PD(S)$ with $\mathcal{P}$ to get a convex tessellation of $\mathcal{P}$, with each resulting cell C_i containing one obstacle. We include the power diagram edges that bound C_i as part of C_i.

The strategy to maintain connectivity is as follows: At any time, let C_u be the cell that contains the user u. The robots in C_u will move according to the strategy presented in Section 2.4.3, and maintain the user's connectivity.

The other robots move to "guard" their regions . Let C_i be a region which does not contain u. We project the user onto the boundary of C_i by finding the closest point in

C_i to u using the Euclidean distance (i.e. we ignore the obstacles).

Let u_i be the *closest* point to u in C_i (see Figure 2.15). Because C_i is convex, u_i cannot move faster than u. Therefore the robots in C_i can maintain u_i's connectivity to b by executing the strategy presented in Section 2.4.3. This guarantees that the user's connectivity is maintained by the connecting arm in C_i as soon as the user enters this cell.

We now bound the number of robots.

Lemma 9. *Let m^* be the number of robots, including the base station, used by any optimal solution to guarantee connectivity between u and b in a convex environment with h circular obstacles. Our strategy uses at most $5hm^*$ robots.*

Proof. By definition, the furthest distance from b to any point in cell C_i is bounded from above by D. Hence the previous bound of $5m^*$ for a single circular obstacle (Theorem 6) applies to each cell C_i. Since the number of cells is equal to h, the number of robots used by our strategy is bounded from above by $5hm^*$. □

Extension to non-circular obstacles

When the obstacles are non-circular, the notion of a radius is undefined and power diagrams cannot be applied as such. However, if we can find an enclosing circle for each obstacle such that the circles are disjoint, it is straightforward to extend the previous result. In certain cases, a tessellation exists even if the disks defined by minimum enclosing circles are intersecting.

Consider obstacles o_i and o_j with enclosing disks s_i and s_j (see Figure 2.16). When $s_i \neq s_j$ intersect each other, but do not intersect o_j and o_i respectively, the power diagram edge $p \in PD(S)$ that is formed by s_i and s_j lies in the region $s_i \cap s_j$, which does not contain any obstacle. The resulting partition cells remain convex [68]. Therefore in this situation we can enact our strategy, and because each cell requires up to $\lceil 5D \rceil$ robots, the upper bound on the number of robots we need for the whole domain is $5hm^*$.

2.4.5 Beyond $O(h)$-Approximation

Can we better approximate the number of robots necessary to maintain connectivity in an arbitrary environment with h obstacles? In this section, we provide further insights

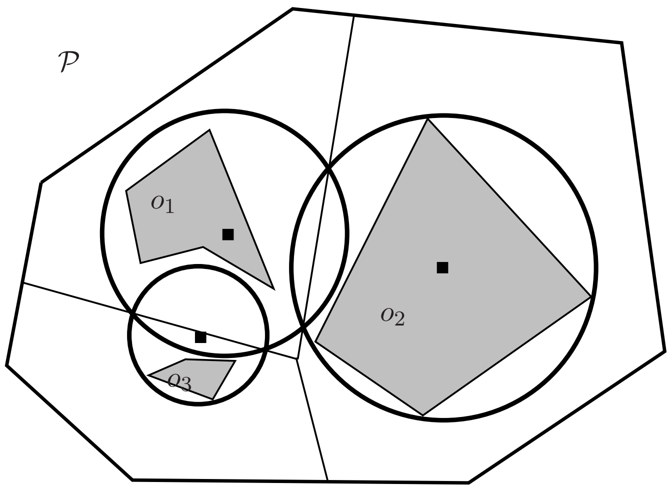


Figure 2.16: An example of a valid tessellation for a domain with three obstacles o_1, o_2, o_3 with enclosing circles s_1, s_2, s_3 also shown. The obstacles are not circles, but each one is entirely contained within a single circle.

on the performance of the algorithm we presented in Section 2.4.4. Recall that in order to compare the performance of our algorithm with respect to the optimal solution, we used a simple lower-bound on the required number of robots based on the parameter D. In many scenarios however this is a very loose lower bound. One way of improving the approximation ratio is to obtain a better lower-bound. Unfortunately, this seems very difficult. In general, the optimal solution can utilize arbitrary topologies and change them dynamically. However, under the following assumptions on the topology of the network, one can obtain better insights on the nature of the performance of our algorithm.

A-1. The topology of the network is an octopus topology which consists of arms directly connected to the base station on one end. The user is connected to the other end.

A-2. The strategies are stationary, i.e. when the user returns to a previously visited location, the configuration of the robots is the same as before.

These assumptions are in fact desirable constraints in practice. An implementation that requires dynamically changing the network topology can have a huge communication overhead. The first assumption A-1 defines a simple yet flexible topology, and clear communication direction for each arm: from the end point (connected to the user u) to the base station b. Only the routing table in the base station will be updated. Assumption A-2 allows the robots to plan their motion based only on current location of u.

Under these assumptions, we are now ready to present two extremal examples which demonstrate the quality of the approximation performance our algorithms. They also shed light on the nature of an optimal solution which can be utilized when designing algorithms.

In the next subsection, we present an instance where the optimal solution needs to use almost as many robots as the number of robots used by our strategy. In these instances our solution is almost optimal. Afterwards, we present an instance where the lower bound on the number of robots used by the optimal strategy is close to D; this will show that our analysis is tight, and the approximation ratio of our algorithms is $\Omega(h)$.

Example-I: $O(h)$ arms may be needed

Consider a regular h-gon $\mathcal{P}_h$ with the base station b at its center. There are h circular obstacles, each of radius r, distributed evenly around b, close to the boundary of $\mathcal{P}_h$. The left side of Figure 2.17 illustrates an instance with $h = 12$. In such an environment, our strategy divides the domain into h regions, allocating $O(D)$ robots for each region where D is the maximum distance from b. This makes a total of $O(Dh)$ robots. The left side of Figure 2.17 shows the power diagram edges, and a wrapping arm and a connecting arm when user u is at the location shown.

We now show how one can construct an environment, for any given h, such that the optimal solution must use $\Omega(Dh)$ robots by generalizing the environment shown in Figure 2.17.

We start with a regular h-gon and place the centers of the obstacles on a circle of radius R centered at b, such that they are equally distributed. At first it appears that our construction has at least three independent parameters: r (size of the obstacles), R (size of the distribution of the obstacles) and D (size of the domain). However, some of these parameters constrain others. They are chosen as follows:

First, the obstacles are taken to be larger than the communication radius so that

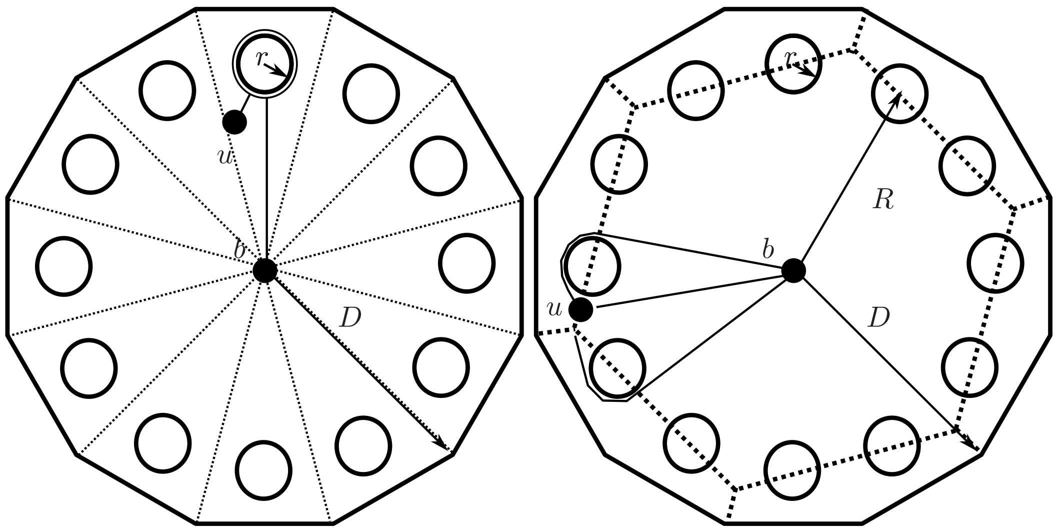


Figure 2.17: The left figure shows the solution given by our strategy. The configuration of the connecting and wrapping arms are illustrated when the user is at the top region. The right figure shows the partitioning of the optimal strategy. When the user is at u the configuration of the arms dedicated to two left regions and central region are illustrated.

the robots in the optimal solution cannot be stationary, i.e. we set r to a constant greater than $1/\pi$. Since we place obstacles at a constant distance from the boundary, fixing R determines D. From now on, we use R and r as the only free variables in the construction.

We now focus on the optimal solution. Any desired strategy that respects both A-1 and A-2 must partition $\mathcal{P}_h$ into regions such that a unique arm is responsible for connecting user u in each region. Further, since two neighboring partitions must be both convex at their common boundary, the partitions must be obtained by slicing the polygon with straight lines. An optimal solution is one which partitions the environment so that the number of robots used is minimized. When there is one obstacle in the environment, the optimal partitioning has two regions as shown in the left part of Figure 2.18. When there are two obstacles (see middle of Figure 2.18), one possible partitioning is determined by the line crossing the centers of the obstacles. Observe that due to the communication range the tip of the arm does not have to go all the way to the line. Hence when u moves from one side of the obstacle to the other side, the movement of the the robot at the tip of the arm will be less. More specifically for any line crossing the communication arcs xy and zt determines a valid partitioning.

When there are three obstacles and obstacles are almost collinear as shown right of Figure 2.18, an optimal solution can use a single line which crosses the communication arcs of all three obstacles. In our construction, we will prevent this case by choosing R accordingly. Hence an optimal partitioning will have at least $h/2 + 1$ partitions. We choose R such that the following constraints are satisfied:

$$(i)\ R \geq \frac{2r\sin(1/r)}{1-\cos(2\pi/h)} \qquad\qquad (ii)\ R \geq \frac{r}{\sin(\pi/h)}$$

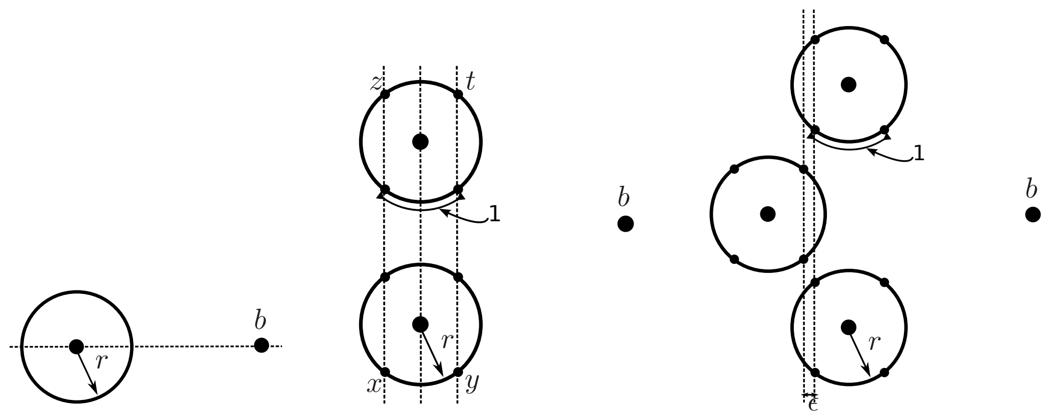


Figure 2.18: When there is only one obstacle, the line connecting b to the center of the obstacle optimally partitions the space, as shown in the left figure. If there are two obstacles, any line crossing the arcs labeled by xy and zt is an optimal partitioning line, as shown in the middle figure. If three obstacles are almost collinear, we can find a line which crosses through the communication arcs shown, as shown in the right figure. In our construction, we prevent this case by choosing R accordingly.

The first constraint ensures that no three obstacles can be separated by a single line, and the second constraint ensures that obstacles do not overlap. When the environment obeys these constraints, there must be a unique partition for every two obstacle. The right of Figure 2.17 shows the optimal partitioning when $h = 12$, and a snapshot of the configuration of three arms are also shown. Using this construction we guarantee that optimal uses at least $hR/2$ robots which is only a constant times better than our strategy.

Example-II: $O(h)$-approximation ratio is tight

Even though the previous example is encouraging, there are some cases where the performance of our approximation is $\Omega(h)$. In the following example, the environment is a $D \times D$ square and h obstacles of radii $r > 2$ are placed at the bottom of the environment. Since there are h obstacles, our solution uses $O(Dh)$ number of robots (see left of Figure 2.19). However, in this case the optimal solution can do better than that. If we partition the environment through the centers of the obstacles, then two arms of length $O(D)$ is sufficient to keep the user connected. This strategy is shown in the right of Figure 2.19. Since the number of robots used by optimal strategy is $O(D)$, our solution is an $O(h)$ approximation in this instance; therefore our analysis is tight.

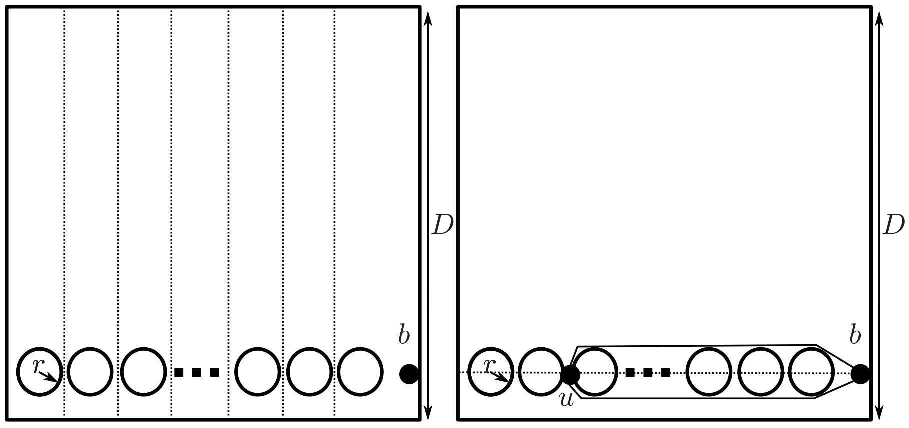


Figure 2.19: When all the h obstacles are collinear with b, our strategy divides the polygon into $O(h)$ regions and uses $O(hD)$ robots, as shown in the left figure. On the other hand, the optimal solution (shown in the right figure) uses only two arms of length $O(D)$ to guarantee connectivity. This shows that our analysis is tight.

2.5 Simulations and Experiments

We demonstrate the practical feasibility of the robotic router systems through simulations and experiments. Our simulations and experiments are designed for creating a robotic router network on the floor where the Robotics Laboratory at Rensselaer Polytechnic Institute (RPI) is located (see Figure 2.20 and Figure 2.21). The Computer Science Department owns only a small portion of this floor (left block) and our wireless network covers only the part of the floor owned by the department. Hence, creating a robotic router network is an appealing solution to control our robots beyond the area limited by our wireless network. We found out the communication model on our floor empirically by measuring the signal between two robots from varying locations (see Figure 2.20).

The algorithms implemented in this section are based on the known and unknown user trajectory algorithms presented in Section 2.3. The approximation algorithms presented in Section 2.4 provide efficient solutions in terms of running time. However, the required number of robots can be a constant time more than the minimum number of robots required. Since we have a limited number of robots, we chose the optimal solutions of known and unknown user algorithms over the approximation algorithms. Although the running time of these algorithms are exponential, their computational complexity is practical for our experiments since we use a small number of robots. Now, we start with simulations demonstrating the advantages of using a robotic router network over a stationary network.

2.5.1 Simulations

We demonstrate a practical application of mobile router networks with simulations for the environment shown in Figure 2.21. We discretize the hallways into discrete locations almost uniformly (some degeneracy exists near the corners of halls). We construct the connectivity table according to the following rule: if the distance between two locations is less than a fixed distance τ, and they are on the same hallway, then these two locations are connected. If two locations are not in the same hallway, their connectivity is based

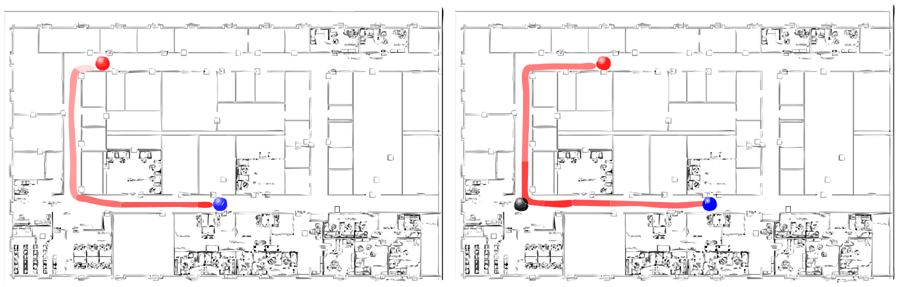


Figure 2.20: The communication model on the floor where experiments take place. **Left:** The blue circle (in the middle of the bottom corridor) is the stationary blue robot. Fading red circles show the signal strength as the red robot moves to the upper corridor (actual measurements). When the red robot reaches the position shown on the map, the signal strength becomes zero. **Right:** By moving a third robot (black circle) to the position shown on the map, we can reestablish the communication. The colors indicate the signal strength from black to red and black to blue nodes.

on the geodesic distance between them [3] . To obtain the connectivity threshold τ' for this case, we subtract a fixed penalty from τ for each turn on the path between the two locations. If the geodesic distance between the two locations is less than τ' then these two locations are connected. In the following simulations, the robotic routers are twice as fast as the user. The base station is located at the bottom of the middle vertical hallway.

In order to obtain a baseline, we computed (by enumeration) the minimum number of static routers to cover the environment. It turns out that at least 4 static routers, as shown in Figure 2.21, are necessary to satisfy coverage and connectivity constraints.

In the following simulations, we start with a network of a single robotic router. For a given (known) user trajectory, we compute the corresponding robot trajectory which keeps the user connected during its trajectory. Next, we find an escape trajectory in which a single robotic router is not sufficient to maintain the connectivity. Finally, we show that two robotic routers are sufficient to keep the user connected whatever

[3] The connectivity model is inspired by the observation shown in Figure 2.20 and is also discussed in Section 2.5.2.

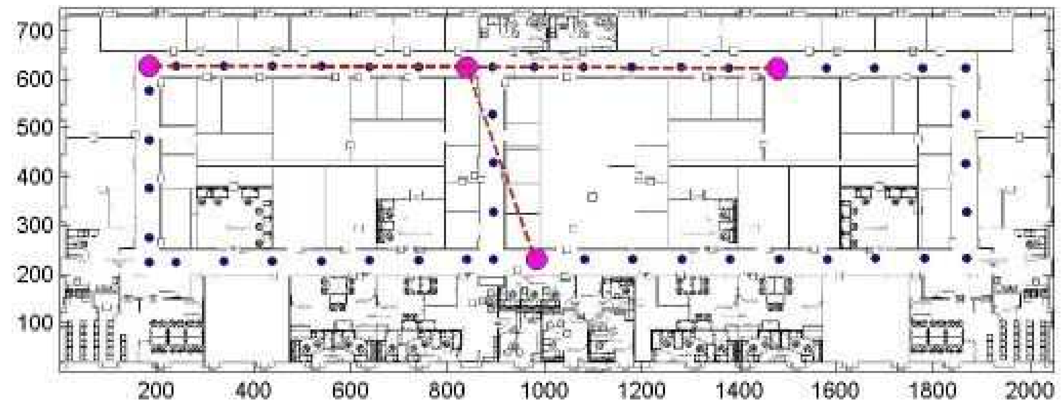


Figure 2.21: The minimum number of static routers to satisfy the connectivity and coverage constraints is 4. The optimum deployment and its network topology is shown.

initial location or trajectory he chooses. We show how two robotic routers keep the user connected even if the user tries to break the connection. Link to the videos of all simulations can be found in Appendix A-Multi Media Extension No: 1.

Figure 2.22 shows the result of our first simulation. The top two figures show the (known) user trajectory and the corresponding robot trajectory computed by our algorithm. We identify the locations of nodes at the critical time steps with time labels. Following figures show snapshots of the connectivity graph of active nodes at these critical time steps. By connectivity graph of active nodes, we indicate the connectivity links (edges) between base station and the user, and the active nodes (vertices) in this connection path.

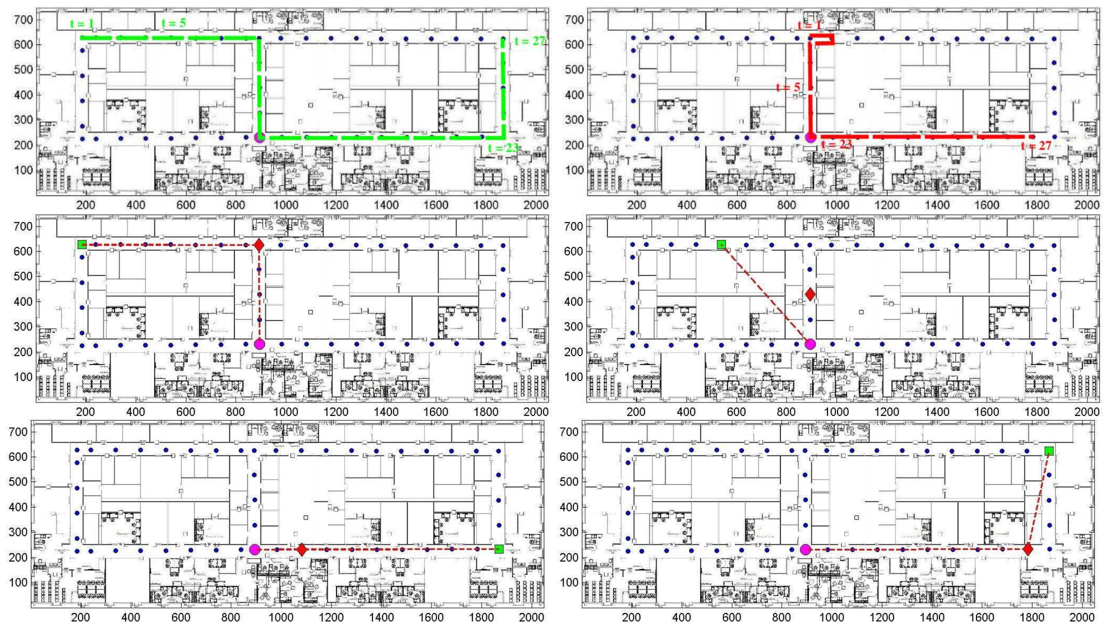


Figure 2.22: The known user trajectory and corresponding computed robot trajectory are shown in top two figures. The remaining figures show snapshots of the user's connectivity graph (base station - circle with magenta color, robotic router - diamond with red color and user - square with green color). The third figure shows the initial configuration of nodes where the user is connected to the base station through the robotic router. The fourth figure shows the configuration at the time step when the user is directly connected to the base station. The fifth figure shows the configuration when the direct link between the user and base station is broken and connectivity is supplied through the robotic router. The last figure shows the final configuration of nodes.

Figure 2.23 shows our second simulation. In this simulation, other than the last turn, the user follows the same trajectory as the previous simulation (see left figure). Until this last turn, the robotic router also follows the same trajectory as the previous simulation. However, in the last step, the robotic router can not keep the user connected. The right figure shows the snapshot at the disconnected state.

We find the minimum number of required robotic routers for all possible user trajectories by trying $AdversarialUserTrajectory$ algorithm with increasing number of robotic routers until there exist corresponding robotic router trajectories for all possible initial locations and trajectories of the user. For this environment, we found that 2 robotic routers are sufficient to provide a continuous connection.

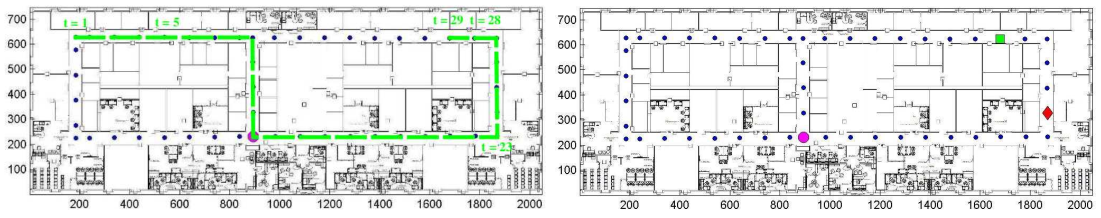


Figure 2.23: Left figure shows an escape trajectory for the user. Until the last turn, the user and the robotic router follow the same trajectory as in Figure 2.22. The right figure shows the snapshot from the last step where the user is disconnected.

In the last simulation, relying on the sufficiency of two robotic routers, we solve the known trajectory algorithm with two robotic routers for a path which is not feasible for single robotic router network. The top three figures in Figure 2.24 show the user trajectory and corresponding robotic router trajectories. The user starts from the top left corner of the environment and completes a cycle by crossing from the middle vertical hall and coming back to the top of the vertical hall. Subsequent figures show connectivity graph of the nodes as snapshots at critical time steps.

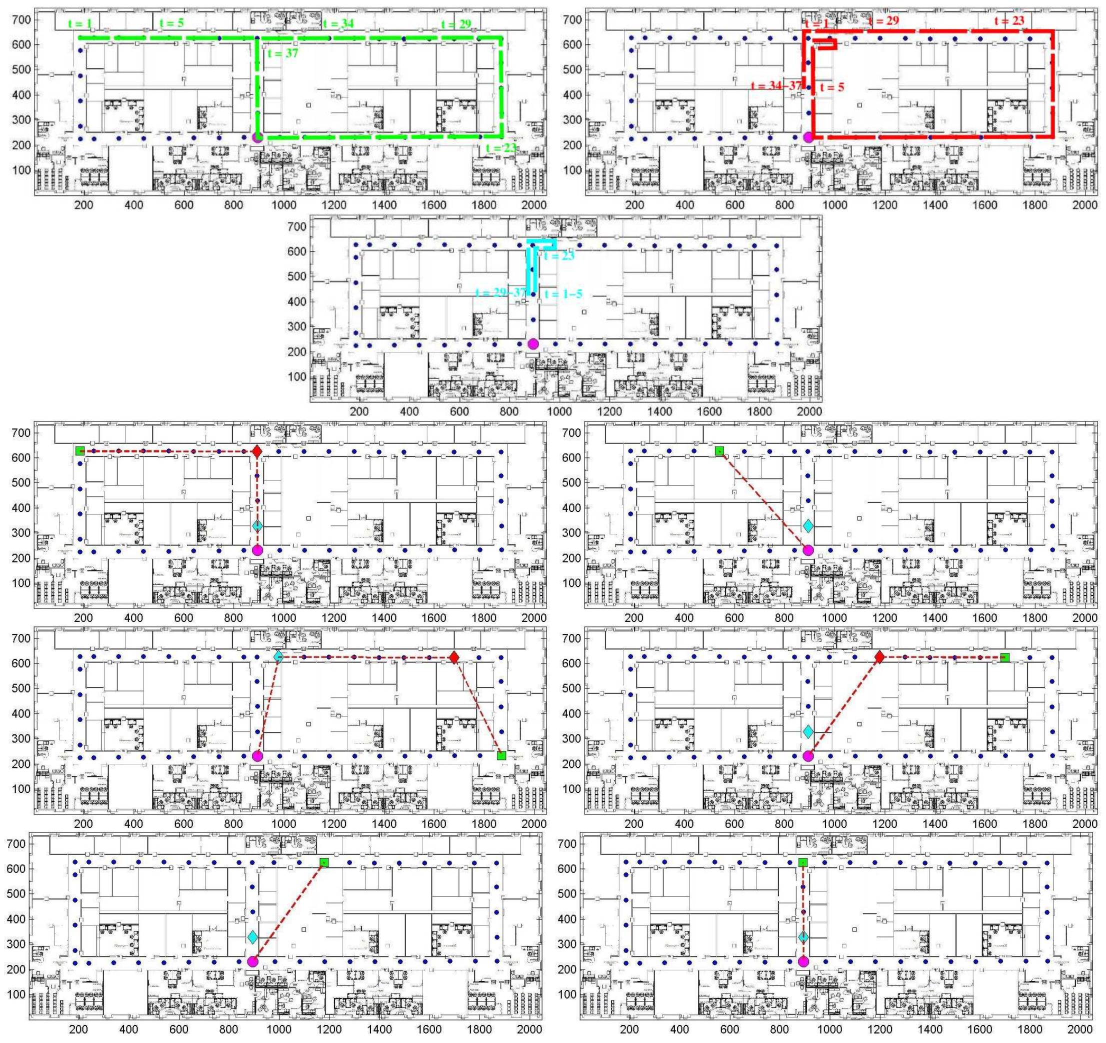


Figure 2.24: Top three figures show the user trajectory and corresponding robot trajectories. Subsequent figures are snapshots from the solution of the algorithm including the connectivity graph of active nodes (base station - circle with magenta color, two robotic routers - diamond with red and cyan color, and user - square with green color). The first figure on the third row shows the initial configurations of all nodes where the user is connected to the base station through the red robotic router. The second figure on the third row shows the configuration at the time step when the user becomes directly connected to the base station. The first figure on the fourth row shows the configuration when the user is connected to base station through three links. The second figure on this row shows the configuration when the three-link connection reduced to a two-link connection. The first figure on the last row shows the time step when the user is directly connected to the base station. The last figure shows the final positions of the nodes at the last time step.

2.5.2 Experiments

To fully test the algorithm presented in Sections 2.3.2 and 2.3.3, experiments were ran in the real world using the robotic router system which is implemented by Wei Yang [69]. Both the known and unknown user trajectory cases were tested to see whether the actual implementation would be able to perform as well as the simulated robots.

Known User Trajectory

In this section, we present an experiment in which the robotic router system maintains the connectivity of the user (a robot), whose trajectory is predetermined, to the base station.

In the experiment, there is a single user robot, two mobile routers and a base station (Figure 2.25). The user starts off at location 1 with the mobile routers at locations 40 and 69, and the base station at location 49, as seen in Figure 2.25. The triangles represent the actual location of the robots, with a blue circle at the center. The circles are their target locations that the motion planning algorithm produces at each step. The square represents the base station. Initially, the user is connected to the base station through the mobile router which is at the middle hallway. In all experiments, the dark lines highlight the connectivity graph.

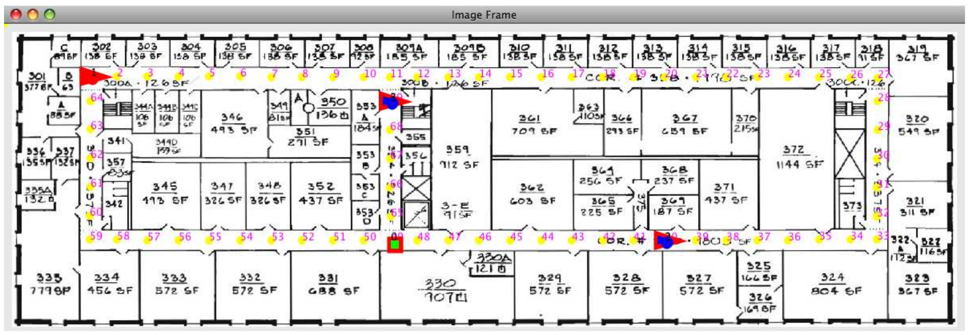


Figure 2.25: An overhead view of the environment for the real world implementation experiments with the initial starting positions of all of the nodes. The triangles represent the actual location of the robots, with a blue circle at the center. The circles are their target locations that the motion planning algorithm produces at each step. The square represents the base station.

The overall experiment proceeded for 15 minutes which corresponds to 10 steps in which the user (solid red triangle) moved down the left hallway and than head right from the bottom hallway. The mobile router at the right of bottom hallway stayed at the same location during the experiment. This is because, the user is never close enough to utilize its services. However, the other mobile router moved up to the top of middle hallway and moved left to maintain the connectivity of the user (second image in Figure 2.26). Several steps of the experiment can also be seen in Figure 2.27 and Figure 2.28.

Although the experiment was relatively short, it was enough to demonstrate that the overall robotic system can implement the formulation of robotic router problem. We used the same connectivity model as we discussed in Section 2.5.1 and this experiment also showed that the connectivity model that we introduced is matched with reality. Moreover, this experiment was able to demonstrate that the implemented robotic routers system can successfully implement the known user trajectory motion planning algorithm from Section 2.3.2. Link to the videos of all experiments can be found in Appendix A-Multi Media Extension No: 1.

Adversarial User Trajectory

In the adversarial user trajectory experiment, the same environment and experiment setup were used as in the known user trajectory experiment. However, the user is a *human holding a laptop* instead of another mobile router. This new user was allowed to move at the same velocity as the robots but its trajectory was not known ahead of time to the base station or the mobile routers. The only requirement for our design is that the user sends his initial location to the base when he requests a network connection. The user then moves by entering commands locally, such as "r" to signal a desire to move right and similarly for the other three directions. The robotic system must then retrieve the user's desired movement before calculating the next location of the mobile routers. For each next location of the user, we consider the user as an adversarial user and choose the motion strategies for robots which maximize the connection time. This ensures a guaranteed performance for a user whose trajectory is unknown.

In the experiment, the user first started off at location 16 (in the top hallway) with the mobile routers and base station all at location 49 (Figure 2.30 and the first image of

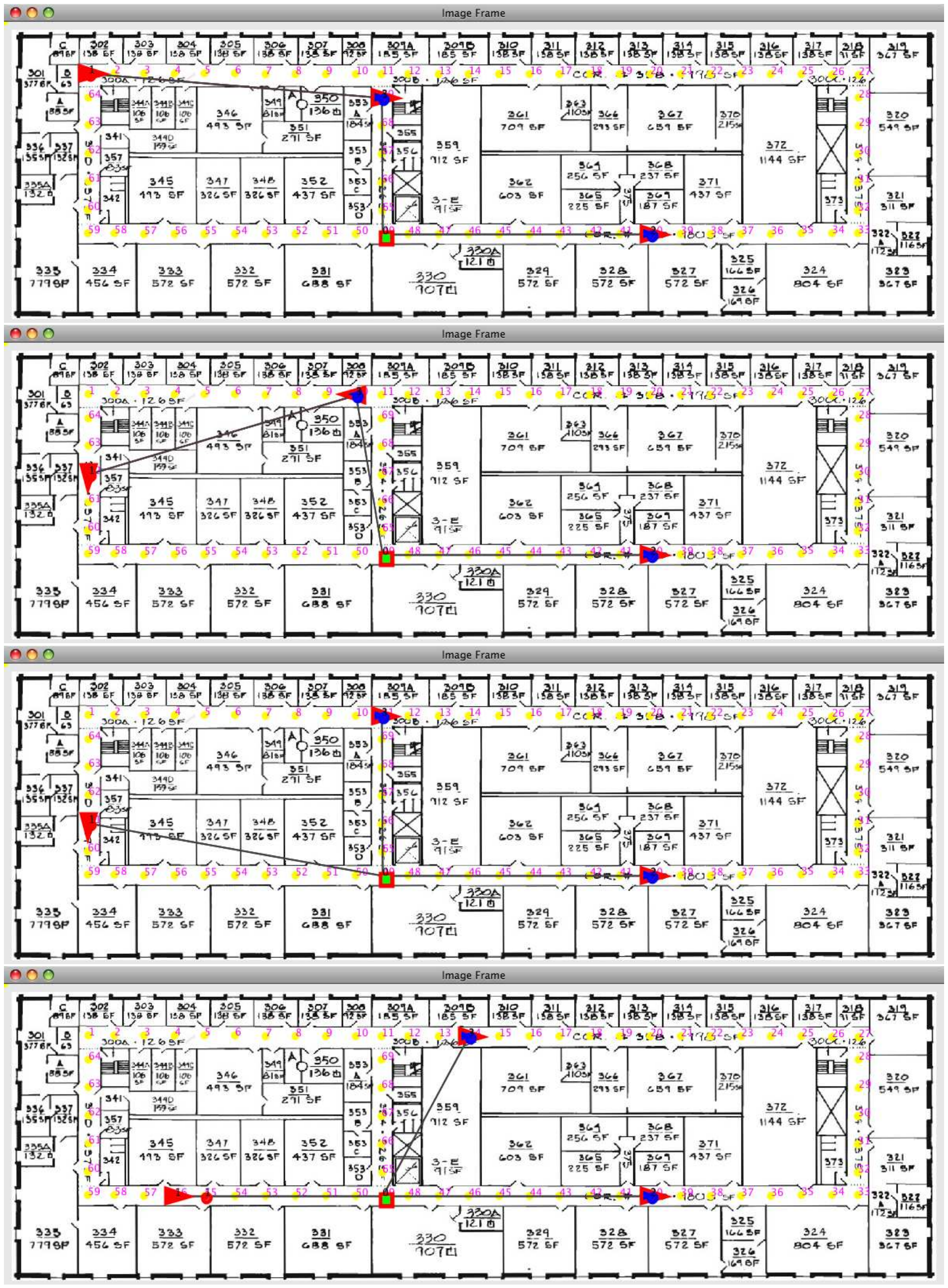


Figure 2.26: This figure shows various stages of the known user trajectory experiment being conducted using the real world robotic system implementation. The dark lines show the connectivity paths between the nodes. Initially, the user is connected through a mobile router but the path changes as the user moves closer to the base station.

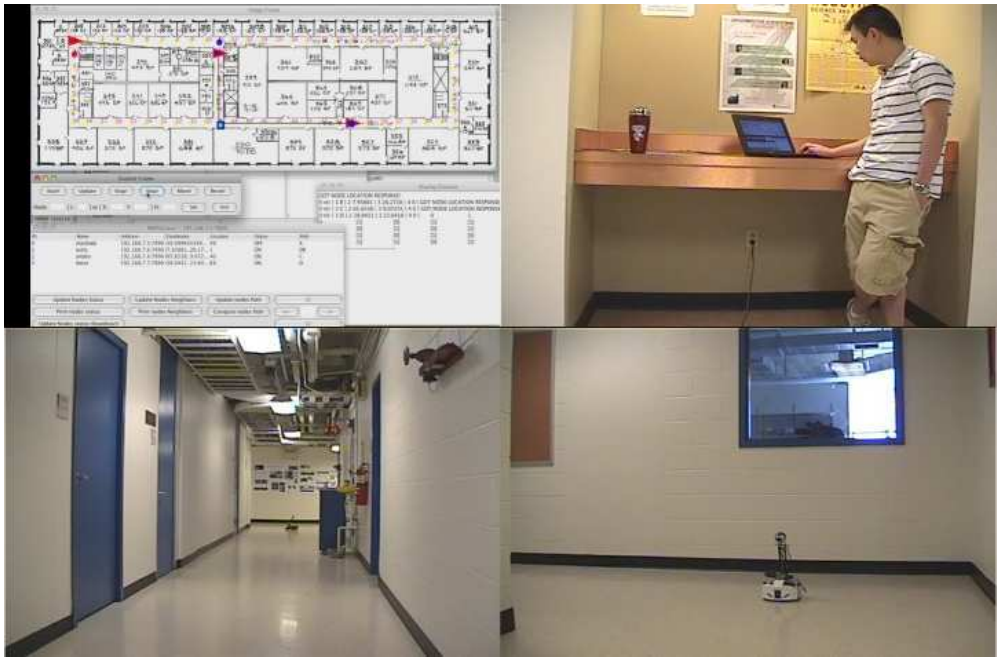


Figure 2.27: **Known user trajectory experiment-1:** Left top figure shows the implementation graphical user interface (GUI). The GUI shows the initial configuration of the robotic router system. Right top figure shows the Mac base station. Bottom left figure shows the mobile router in the middle vertical hallway. Bottom right figure shows the user which is a robot that controlled remotely by the base station.

Figure 2.29). From here, the user proceeded to move right to the right corner of upper hallway. Initially, the mobile routers did not move because the user was still in range of the base station. However, once the user moved to the limits of its communication range, the first mobile router started to move up to relay messages as seen in Figure 2.31 and second image of Figure 2.29. As the user continued to move right, the first mobile router also continued to move up, using itself to maintain the user to base connection. Once this mobile router reached top of middle hallway, it stopped moving because the user was now visible and connection range was extended. The connection was maintained from mobile routers current position until the user reached to the right end of upper hallway. When user was at the end of hallway mobile router moved one step right to maintain the connectivity (the last image of Figure 4.33). During this experiment, we

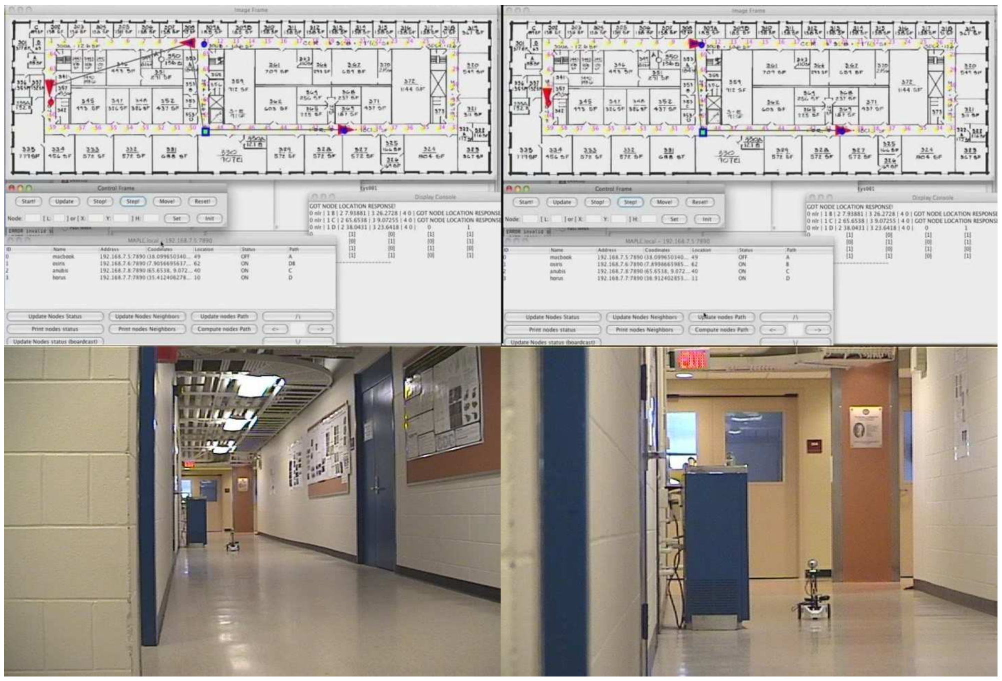


Figure 2.28: **Known user trajectory experiment-2:** This figure shows the time when the connection path of the user is changed as it moves down the middle of the left hallway. The top row shows the configuration of mobile router network on the GUI while the bottom row shows the corresponding user location at that time step. The left column shows the final time step when user is connected to base station through the mobile router. The right column shows the time step right after the direct connection of user to base station is satisfied.

acquired perfect connection between the user and base station, and user did not lose any sent data packets.

From this second experiment, the system clearly demonstrated the ability to react dynamically to an unknown user trajectory. Since, we do not know the trajectory of the user, we react to the user as it is an adversarial user. The trajectories of robots was dynamically calculated using the table extracted by *AdversarialUserTrajectory* algorithm. Our implementation chose the robotic router strategies which maximizes the connection time of the user according to this table. Hence, the robotic system successfully maintained the connectivity of the user by moving the mobile routers to their optimal

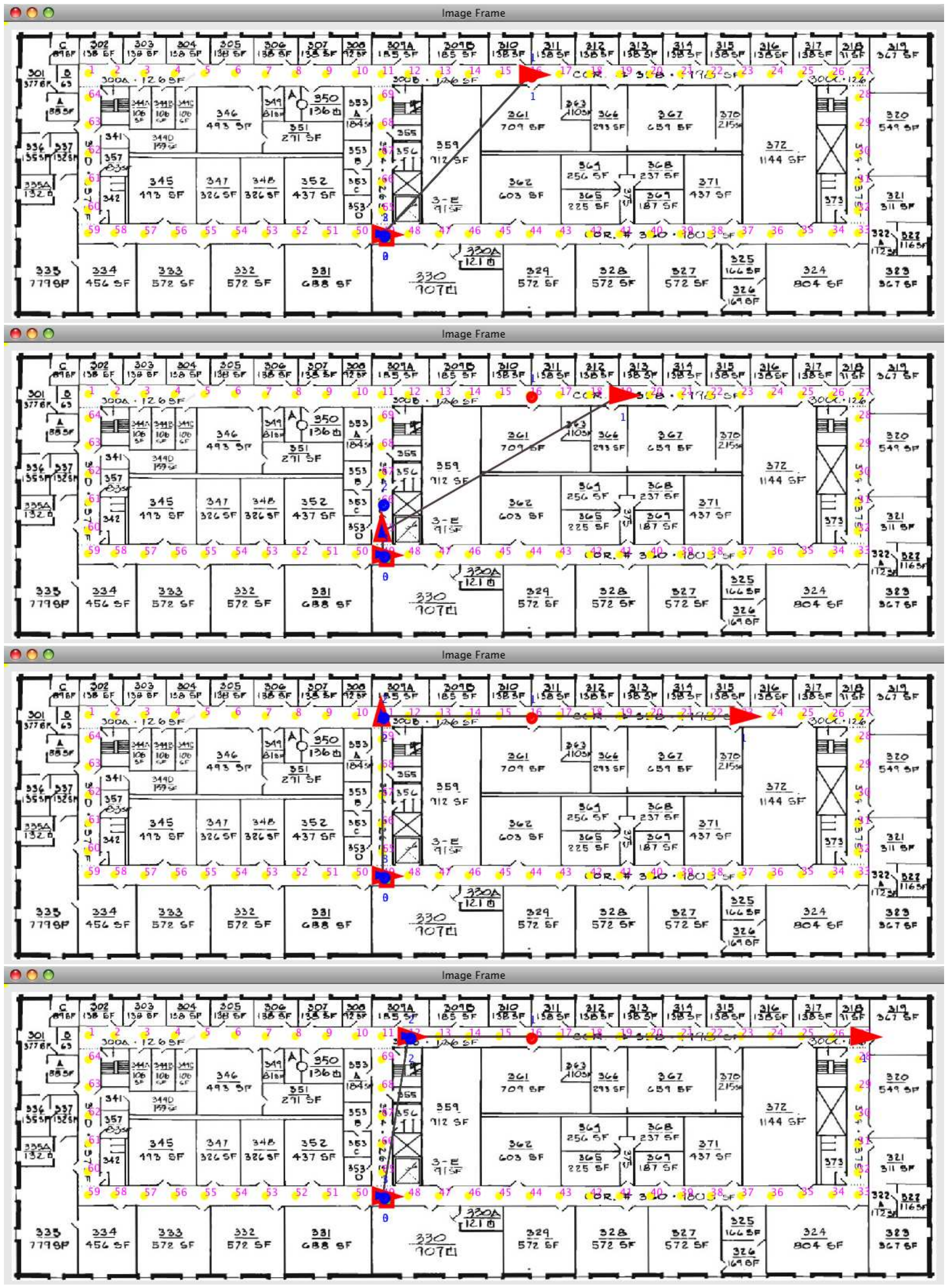
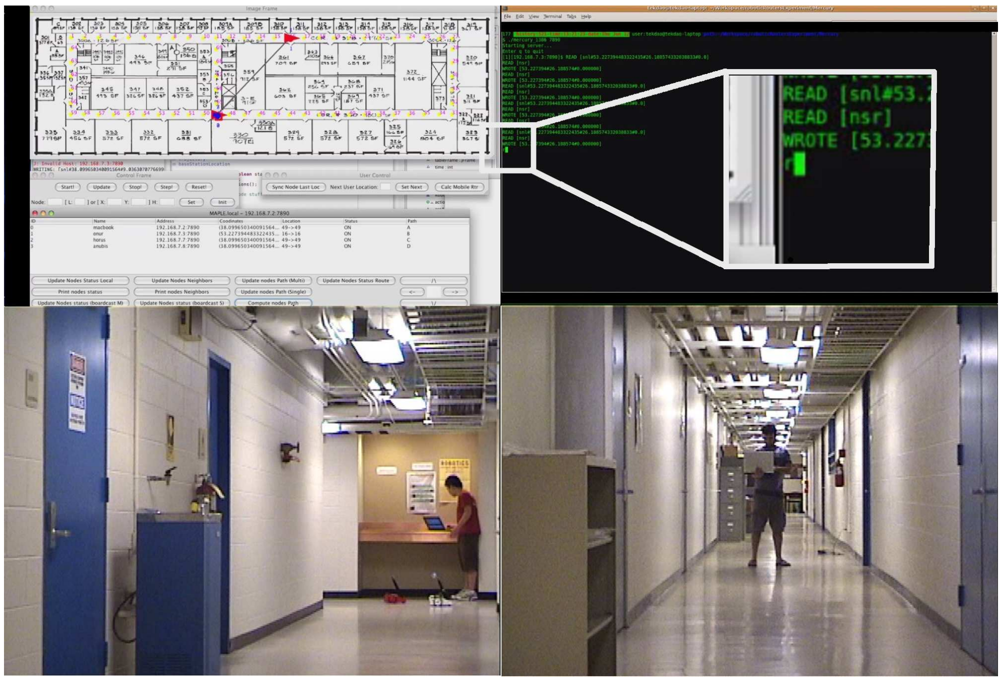


Figure 2.29: This figure shows various stages of the adversarial user trajectory experiment being conducted using the real world robotic system implementation. The dark lines show the connectivity paths between the nodes. The first figure shows the initial configuration of the mobile router network. The second screen shows a mobile router moving up to maintain connectivity as the user moves to the right. After reaching the top, the mobile router remains stationary until the user reaches the end of the hallway. When this occurs, the router starts to move right in an attempt to maintain connectivity between the user and base station.



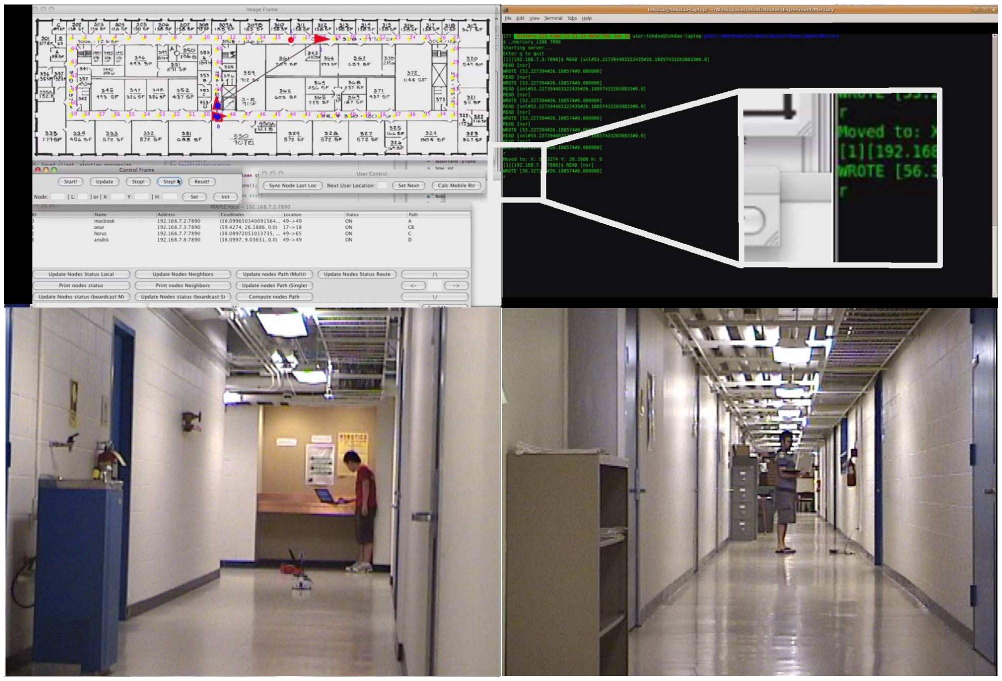
```
$ ./mercury_1386 7890
Starting server...
Enter q to quit
[1][192.168.7.3:7890]$ READ [snl#53.227394483322435#26.188574332038833#0.0]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [snl#53.227394483322435#26.188574332038833#0.0]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [snl#53.227394483322435#26.18857[illegible]33#0.0]
READ [nsr]
WROTE [53.227394#26.[illegible]000000]
READ [nsr]
[illegible]#26.188574#0.000000]
READ [snl#[illegible].227394483322435#26.188574332038833#0.0]
READ [nsr]
WROTE [53.[illegible]7394#26.188574#0.000000]
r
```

Figure 2.30: **Unknown user trajectory experiment-1:** This figure shows the initial locations of the mobile router network. In this experiment, the mobile router network keeps the connectivity of an adversarial user (laptop) who requests wireless connectivity and sends acknowledgment of its move in each time step. The top left figure shows the GUI, the top right figure shows the message sent from user, the bottom left figure shows the robots and base station, and the bottom right figure shows the user.

locations.

2.6 Concluding Remarks

In this chapter, we addressed the problem of creating an adaptive network between a stationary base station and a mobile target. In this problem, robotic routers are used to create a communication bridge between the base and the target. For arbitrary communication models, we presented two optimal strategies for two user motion models. In the first model, the user's trajectory is known whereas in the second model the user moves in an adversarial fashion. Even though the algorithms compute optimal solutions,



MAPLE.local – 192.168.7.2:7890

ID	Name	Address	Coodinates	Location	Status	Path
0	macbook	192.168.7.2:7890	(38.099650340091564...	49->49	ON	A
1	onur	192.168.7.3:7890	(59.4274, 26.1886, 0.0)	17->18	ON	CB
2	horus	192.168.7.7:7890	(38.08972051013735, ...	49->65	ON	C
3	anubis	192.168.7.8:7890	(38.0997, 9.03631, 0.0)	49->49	ON	D

```
tekdao@tekdao-laptop: ~/Workspace/roboticRo
177 user:tekdao@tekdao-laptop
$ ./mercury_i386 7890
Starting server...
Enter q to quit
[1][192.168.7.3:7890]$ READ [snl#53.2273944833224 35#26.188574332038833#0.0]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [snl#53.227394483322435#26.188574332038833#0.0]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [snl#53.227394483322435#26.188574332038833#0.0]
READ [nsr]
WROTE [53.227394#26.188574#0.000000]
READ [nsr]
WROTE [53.227394#26.188574#0.0
READ [snl#53.22739448         .188574332038833#0.0]
READ [nsr]
WROTE [53.22     26.188574#0.000000]
r
Moved to: X: 5 .3274 Y: 26.1886 H: 0
[1][192.168.7. 7890]$ READ [nsr]
WROTE [56.327  #26.188574#0.000000]
r
```

Figure 2.31: **Unknown user trajectory experiment-2:** This figure shows the second step of the user. The user continues his movement in the right direction (the input “r” shown in top right figure). To keep him connected, the mobile router moves one step forward.

their running times are exponential in the number of robots in the network. We improved the running time of the previous algorithms by presenting an approximation algorithm for a geometric instance of the robotic router problem. In this formulation, two entities can communicate if the geodesic distance between them is less than a threshold. We presented an optimal (in terms of the number of routers) algorithm for simply-connected polygons, a constant factor algorithm for a polygonal environment with a single obstacle, and an $O(h)$-approximation algorithm for environments with h obstacles. Finally, we demonstrated the practical feasibility of robotic router systems through simulations and experiments.

Acknowledgment

The results of this chapter appeared in ICRA2008 [19], ICRA2010 [70] and the International Journal of Robotics Research (IJRR) [71]. This work is joint with Nikhil Karnad, Patrick Plonski, Wei Yang and Prof. Volkan Isler.

Chapter 3

Communication Bridge

The task of building a communication bridge connecting two locations arises frequently. For example, when fighting forest fires, a high capacity connection between the command center and a temporary base may be needed. When there is no underlying communication infrastructure (which is typically the case in emergency response scenarios), mobile entities with communication capabilities can be used to build a communication bridge. In particular, with recent advances in robotics, using mobile robots for this purpose is becoming feasible.

In this chapter, we address the problem of building a communication bridge in an efficient fashion. Imagine that we are given a source s and a destination t (the two locations that need to be connected), and initial locations of n robots (or *mobile hubs*). The goal is to pick a small subset of these robots and determine their final locations, so that when the robots arrive at their final locations, there is a path between s and t in the underlying communication graph. In this case, we say that a *communication bridge* between s and t has been established. Throughout the chapter, we assume that two entities can communicate if and only if they are within a given communication radius r. See also Figure 3.1.

We focus on two measures of efficiency. The first one is the distance traveled by the robots to establish the communication bridge. Relevant objectives are minimizing the maximum or the total Euclidean (L_2) distance traveled. This measure is important when the robots have limited battery power. The maximum distance traveled also determines how quickly the bridge can be established. The second measure is the number of robots

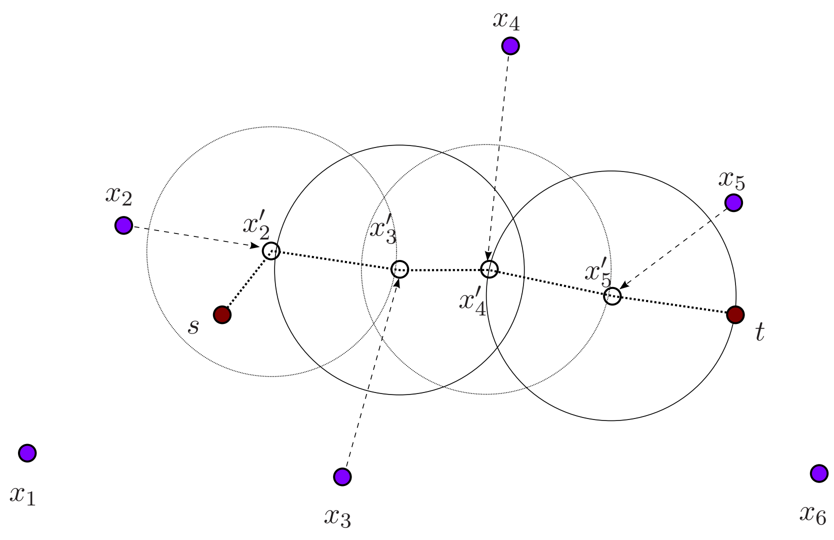


Figure 3.1: Initial locations of the robots are x_i, $i = 1, \ldots, 6$; s and t cannot communicate. By moving $x_2 \rightarrow x'_2$, $x_3 \rightarrow x'_3$, $x_4 \rightarrow x'_4$ and $x_5 \rightarrow x'_5$ a communication bridge with four hubs connecting s and t is established. The circles around the nodes illustrate their communication radii.

required to establish the communication bridge. This is an important parameter because if we use a small number of robots for the given task, then the remaining robots can be used for other tasks. In addition, a communication bridge with a small number of hubs is desirable in order to minimize the latency of the network.

Our results and techniques. We believe that the two metrics mentioned above are equally important. Therefore, we include both metrics in the optimization problem we study. More specifically, we present algorithms to minimize the number of hub robots and their maximum (or total) movement to create a communication bridge between two stationary locations.

The general problem where the environment is represented with an arbitrary graph is NP-hard and, in fact, cannot be approximated efficiently [23]. . Hence, in this chapter, we focus on a geometric version where the underlying environment is the Euclidean plane, and the chosen robots are required to move onto the straight line segment $[s, t]$ to form a communication bridge. This special case is important from a practical standpoint because moving the robots onto this line segment yields the minimum number of hubs in the communication bridge, as compared to any other curve joining s and t. Another motivation for this model is low power, inexpensive infra-red communication which is becoming a popular choice for small robots: In an extreme case, if each robot is equipped with only two IR receivers/transmitters such that the pairs are placed 180 degrees apart, a straight line communication is necessary to establish a communication bridge between s and t. From a theoretical perspective, these problems turn out to be quite challenging. One of the major sources of difficulty is the lack of an "ordering property" in the optimal solution (We make the ordering property explicit in Section 3.2.1.) As an example, consider the version where we are given a maximum travel distance for each robot. Suppose robot a (resp. robot b) can reach points inside the line segment $[l_a, r_a]$ (resp. $[l_b, r_b]$). It is possible to build instances where r_a is to the left of r_b but in the optimal solution robot a moves to the right of robot b (Figure 3.2).

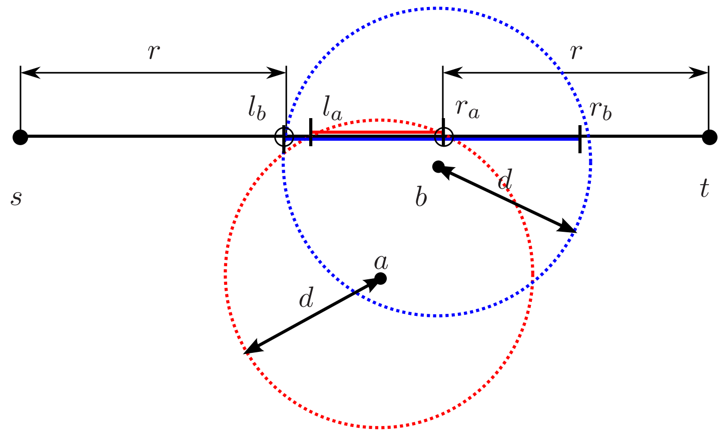


Figure 3.2: Robot a (resp. robot b) can reach points inside the line segment $[l_a, r_a]$ (resp. segment $[l_b, r_b]$). Although a is to the left of b, a must move to the right of b, to r_a, and b must move to the left of a, to l_b, to establish a communication bridge. The final locations of robots are shown by unfilled circles.

For the maximum distance version (*MaxDist*), we overcome this hurdle by relaxing the distance requirement: if the optimal algorithm can build a communication bridge with at most k hubs by moving each robot at most distance d, we present an approximation algorithm which builds a communication bridge with k hubs by moving each robot at most distance $\sqrt{2}d$ (Section 3.2.1). The key result enabling the algorithm is the presence of an ordering property for the relaxed version. For the sum version (*SumDist*), we show that there is an ordering property but for the L_1 metric (sum of absolute values of coordinate differences). We present an algorithm which exploits this ordering property and returns the optimal solution for the L_1 metric. This in turn yields a $\sqrt{2}$-approximation algorithm for the L_2 case (Section 3.2.2).

The algorithms we present are dynamic programming solutions which exploit the ordering property. However, even with the ordering property, the dynamic programming solutions are not straight-forward. This is mainly because the final locations of the robots must be chosen from the continuous set of points on the line segment $[s, t]$: There are instances in which robots must be placed precisely to achieve the optimal solution, and slightly perturbing the optimal solution (to a finite set of points) breaks connectivity. Therefore, our algorithms avoid an a priori discretization of the line segment.

Finally, we present an interesting property regarding the number of hubs. Let $L > r$ be the distance between s and t. Clearly, at least $n^* = \lceil L/r \rceil - 1$ hubs are required to connect s and t. However, building a bridge with n^* hubs may not be feasible due to the motion constraint. We show that any minimal solution which satisfies the motion constraint uses at most $2n^*$ hubs (Section 3.3). This means that by removing constraints on distance we can gain a factor of at most 2 in the number of hubs.

3.1 Related Work

Exploiting the controlled mobility of robots to create a connected network has received significant attention in robotics community. In the coalescence problem, the robots are scattered in an environment and their locations are unknown to each other. The goal is to create a single connected network [42]. In [72], the authors present a strategy where each robot performs a random walk. When robots meet they create a connected cluster and continue the same strategy as a single robot. Authors analyze the coalescence time for the random walk strategy. In rendezvous problem, the goal is to minimize the time required for two or more players (robots) to meet [41]. The robots are assumed to have unit speed and they are randomly deployed in an environment. This problem is extensively studied in the literature. Depending on whether or not players can decide their strategies in advance, rendezvous problem is studied under two classes. In symmetric rendezvous [73–76] problem, each player has to execute the same strategy. In the asymmetric version of the problem [41, 77, 78], players can decide their roles in advance, e.g. while one player waits, the other player can be charged for searching the other robot.

In the related, *Network Formation Problem (NFP)*, robots whose locations are unknown are placed in an environment . Initially, a single robot has the information. This robot searches for other robots to share the information. Once the information is shared, other robots can participate for the information propagation task [79]. A related problem to NFP is the *Freeze-Tag Problem (FTP)* [80]. In FTP there is a single awakened robot and all the other robots are asleep (frozen). A robot can be awakened only when an awakened robot visits (tags) it. The objective is to awake all the robots as early as possible [81].

In the previous work, network formation problems are formalized as creating a single network from mobile robots. On the other hand, in some applications, a network has to be established in according to accomplish a specific task such as creating a communication bridge between two stationary locations. In this work, we address the problem of creating a communication bridge while minimizing the number of robots and their movement.

In [23], Demaine et al. studied the problem of moving pebbles along the edges of a graph (with n vertices) so as to achieve various connectivity objectives while minimizing the number of moves. In particular, they sketch an $O(n)$-approximation algorithm for the problem of creating a path of pebbles between two given vertices with minimum sum distance. They also show that minimizing the total or maximum distance is NP-hard, and that the maximum distance case cannot be approximated within a factor $\Omega(n^{1-\epsilon})$. Since connectivity and mobility are coupled in their model, their results do not directly apply to the problems studied here. In this chapter, we present the first results for the problem of building a communication bridge while minimizing the number of hubs and the distance traveled by them for a given communication radius.

3.2 Building a Bridge with the Minimum Number of Hubs

In this section, we study the problem of building a communication bridge between s and t while optimizing the number of hubs and the movement of the robots. We present solutions to two bi-criteria optimization problems: In the first problem (*MaxDist*), we seek a solution with the minimum number of hubs subject to the constraint that each robot moves at most a given distance d. In the second problem (*SumDist*), the constraint is that the total movement must not exceed B. In this chapter, we present algorithms for given d or B. To find the minimum value of d (resp. B), one can perform a binary search on d (resp. B resp.).

A couple of remarks: When the distance between s and t is less than r, i.e. $|st| \leq r$, there is no need for any intermediate robots. Hence, we consider the case where $|st| > r$. Also, in order to achieve a bridge between s and t, it is both necessary and sufficient that the distance along $[s, t]$ between every consecutive pair in the communication bridge is at most r. Therefore, $|st| \leq (n+1)r$ holds. Hence, we assume that the number of robots n is at least $\lceil |st|/r \rceil - 1$.

3.2.1 MaxDist: Minimizing Maximum Distance

In *MaxDist*, we are given points s and t and a set, $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, of point-robots in the plane and a maximum traveling distance d. Any two members of $\mathcal{P} \cup \{s, t\}$ can communicate with one another if they are within (Euclidean) distance r of each other. Let $u_i = (x_i, y_i)$ be the initial position of p_i. We wish to select a subset $S \subseteq \mathcal{P}$ and compute a final position $v_i = (x'_i, y'_i)$ on the line segment $[s, t]$ for each $p_i \in S$ such that (i) s and t are connected via point-to-point communication links where points are selected from the final locations of robots in S and link lengths are not greater than the communication distance r, (ii) the distance traveled by each robot p_i is not greater than d (i.e. $\forall_{p_i \in S} |u_i v_i| \leq d$), and (iii) the total number of hubs in the communication bridge (i.e. $|S|$) is minimized.

Let L be the line passing through s and t. We place a coordinate frame where the x-axis is aligned with L, s is at 0 (i.e. $x_s = 0$) and t is at location $x_t > 0$. Without loss of generality, we define *right* as the positive direction of this frame. The final location of robots $p_i \in S$ can be determined as $v_i = (x'_i, 0)$ in this new coordinate frame. Hence, we can use x'_i to denote the final location $v_i = (x'_i, 0)$. Also note that the projection of the initial location $u_i = (x_i, y_i)$ on to L is simply x_i.

We start by pruning the set $\mathcal{P}$ and removing robots which are more than distance d away from L (i.e. if $|y_i| > d$ then p_i is removed). Moreover, we can remove the robots

p_i such that $x_i < -d$ or $x_i > x_t + d$. This is because these robots cannot reach the line segment $[s, t]$. Let us call the new set which consist of robots satisfying the above constraints as $\mathcal{P}'$.

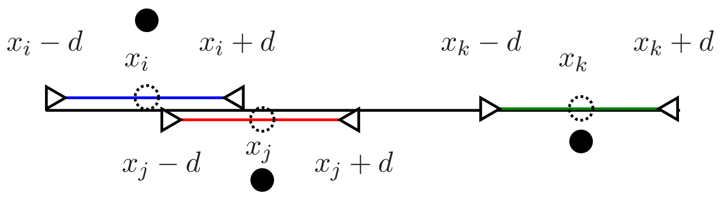


Figure 3.3: Let x_i be the projection of the initial location of robot p_i. We relax the final location of p_i to $l_i : [x_i - d, x_i + d]$ which is shown as the left-most line segment.

For each robot $p_i \in \mathcal{P}'$, we compute a line segment $l_i : [x_i - d, x_i + d]$ (Figure 3.3). We will pick the final location of p_i from this line segment. Note that this is a relaxation because the robot may have to move more than distance d. But the deviation is bounded as it is stated in the following proposition:

Proposition 10. *For any final location $x'_i \in [x_i - d, x_i + d]$ where $p_i \in \mathcal{P}'$, the distance traveled is not greater than $\sqrt{2}d$, i.e. $|u_i v_i| \leq \sqrt{2}d$.*

Proof. The maximum distance traveled for p_i is $\sqrt{d^2 + y_i^2}$ when the movement is relaxed to l_i. Since $y_i \leq d$ holds, the claim follows, i.e., $\sqrt{d^2 + y_i^2} \leq \sqrt{2}d$. □

The number of hubs required for the relaxed version is not more that the number of hubs required for the original problem:

Proposition 11. *Let k^* and k be the number of hubs used in an optimal solution to the original problem and an optimal solution to the relaxed problem, respectively. Then, $k \leq k^*$.*

Proof. An optimal solution to the original problem cannot place a robot p_i outside of $l_i : [x_i - d, x_i + d]$. Because otherwise the distance traveled in x-direction exceeds the distance constraint d. Hence an optimal solution to the original problem is also a solution for the relaxed case, and k cannot exceed k^*. □

The relaxed version of the problem satisfies a simple ordering property which allows us to design an efficient algorithm. As mentioned previously (Figure 3.2) the original problem may not have the ordering property. We now explain the ordering property satisfied in the relaxed version.

Consider a placement of robots on L where the final location of each robot p_i is chosen from the line segment $[x_i - d, x_i + d]$. We order the robots according to x_i values in non-decreasing order. We say that the placement is *well-ordered* if for any two robots p_i and p_j such that $x_i \leq x_j$, we have $x'_i \leq x'_j$.

Lemma 12 (Ordering Property)**.** *There exists a well-ordered optimal solution for the relaxed problem.*

Proof. In an optimal placement, let us call $(p_i,\ p_j)$ an *unordered consecutive pair* if two robots p_i and p_j which are consecutive in the final bridge, are placed at respective locations x'_i and x'_j with $x_i \leq x_j$ but $x'_i > x'_j$. We claim that there is an optimal solution with zero unordered consecutive pairs. Consider an optimal solution which has the minimum number of unordered pairs. Suppose that this number is non-zero. Let p_i and p_j be two robots forming a consecutive unordered pair (if an unordered pair exists, so does a consecutive one). We show that the final locations of these two robots can be swapped, reducing the number of unordered pairs by one. This contradicts with the minimality of the number of unordered pairs.

First, from the relaxed segment assumption (i) $x'_i \leq x_i + d$ and (ii) $x_j - d \leq x'_j$ holds. Since this is an unordered pair, we have: (iii) $x_i \leq x_j$ and (iv) $x'_i > x'_j$. From (i)-(iv) we have: $x_i - d \leq x_j - d \leq x'_j < x'_i \leq x_i + d$. Observe that $x_i - d \leq x'_j < x_i + d$ holds, hence we can move p_i to x'_j which is in its feasible region.

Similarly, we find that $x_j - d \leq x'_j < x'_i \leq x_i + d \leq x_j + d$. Hence, x'_i is in the feasible region of p_j which makes it possible to move p_j to x'_i.

Finally, we can conclude that we can swap the final locations of p_i and p_j and decrease the number of unordered pairs by one while p_i and p_j remain in their respective feasible regions. Moreover, since p_i and p_j are consecutive, swapping does not introduce additional unordered pairs. This contradicts the fact that the solution has the minimum number of unordered pairs. $\square$

The ordering property allows us to use dynamic programming to compute an optimal solution. Before presenting the algorithm, we define the *reach* of a solution $S = \{p_1, p_2, \ldots, p_m\}$. Without lost of generality, let us assume that S is sorted in increasing order. If there is a communication bridge between s and p_m, then we have a reachable region from 0 to $x'_m + r$ where we can place a robot connected to s. As we assume that reach starts from 0, we can define the *reach* of S with a single parameter, i.e. $reach(S) = x'_m + r$.

Let $OPT(k, i)$ be the maximum *reach* which uses k robots from the set $\{p_1, p_2, \ldots, p_i\}$ to form a connected set with s where $\forall_{1 \le j \le i}\, p_j \in \mathcal{P}'$. To simplify the notation, we define the function $conn(k, i)$. This function returns true if and only if $[x_i - d, x_i + d]$ intersects with the *reach* of $OPT(k, i-1)$. In other words, this function tests whether a robot x_i can extend the *reach* $OPT(k, i-1)$ by moving inside its feasible region l_i and extend the *reach* of s. This condition is satisfied if the following holds: $OPT(k, i-1) \ge x_i - d$ and $x_i + d \ge 0$.

We now present the dynamic programming algorithm.

$$OPT(k, i) = 0 \quad if\ i < k \tag{3.1}$$

$$OPT(0, i) = r \tag{3.2}$$

$$OPT(k, i) = \begin{cases} \min(x_i + d, OPT(k-1, i-1)) + r & if\ conn(k-1, i) \\ OPT(k, i-1) & \text{o/w} \end{cases} \tag{3.3}$$

Since $OPT(k, i)$ uses k robots from the set $\{p_1, p_2, \ldots, p_i\}$ the cardinality of this set cannot be less than k. This condition is addressed by the first equation. The second equation constitutes the base case. When we do not use any robots (i.e. $k = 0$) then the *reach* is r which is the reachability region of s.

In the last equation, we compute all remaining entries $OPT(k, i)$. We know that the optimal solution chooses one of the $j \le i$ as the k^{th} hub. We consider two cases: (1) the last hub is p_i: we look up the optimal solution with $k-1$ hubs which are selected from the set $\{p_1, p_2, \ldots, p_{i-1}\}$. If $[x_i - d, x_i + d]$ intersects with $OPT(k-1, i-1)$ then the optimum solution will put p_i to the rightmost possible location which is $x'_i = \min(x_i + d, OPT(k-1, i-1))$ and we set the *reach* $OPT(k, i) = x'_i + r$. (2) The last hub is not p_i: Then the k^{th} hub should be selected from set $\{p_1, p_2, \ldots, p_{i-1}\}$ whose maximum value is calculated by $OPT(k, i-1)$ in the previous iterations. If the first case suffices, we pick it since it extends *reach* more than the second case (due to the ordering property) otherwise we pick the second case and set it to $OPT(k, i)$.

Using the above formula, we calculate the dynamic programming table where both k and i vary between 0 and m where $m \leq n$ is the cardinality of pruned set $\mathcal{P}'$. From this table we find the minimum k such that $OPT(k, m) \geq x_t$. This yields the optimal solution to the relaxed problem. By Proposition 10, our solution gives a $\sqrt{2}$ approximation on the maximum distance traveled by using at most the same number of hubs used in the optimal solution (due to Proposition 11).

The running time of our algorithm is $O(n^2)$. This is because the size of the table is $O(n^2)$ and for each entry we take the maximum of two values (Equation 3.3).

Theorem 13. *If there exists a solution to MaxDist that uses k hubs such that each robot moves at most distance d, then we can compute a solution where we use at most k hubs and each hub moves at most $\sqrt{2}d$ in $O(n^2)$ time.*

3.2.2 SumDist: Minimizing the Total Distance

In *SumDist*, we are given points s and t and a set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of mobile hubs, as well as a budget B on the total distance traveled. Let $u_i = (x_i, y_i)$ be the initial position of p_i on the plane. We wish to select a subset $S \subseteq \mathcal{P}$ and compute a final position $v_i = (x'_i, y'_i)$ on the line segment $[s, t]$ for each $p_i \in S$ such that (i) s and t are connected via point-to-point communication links, (ii) the total L_2 (Euclidean) distance traveled is not greater than B (i.e. $\sum_{p_i \in S} |u_i v_i| \leq B$), and (iii) the total number of hubs in the communication bridge (i.e. $|S|$) is minimized.

Similar to *MaxDist*, we place a coordinate frame where the x-axis is aligned with L (the line passing through s and t), s is at $x = 0$ and t is at $x_t > 0$. The *reach* of a solution is defined as before.

Unfortunately, there exist instances where the ordering property does not hold in the L_2 metric. However, it turns out that when the underlying distance metric is L_1, there is an optimal solution which satisfies an ordering property, which in turn enables a dynamic programming based solution. We say that a placement is *well-ordered* if for any two robots p_i and p_j such that $x_i \leq x_j$ we have $x'_i \leq x'_j$.

Lemma 14. *If the distance metric is L_1, then there exists a well-ordered optimal solution.*

Proof. Let us assume that OPT_1^* is an optimal solution which includes the least number of unordered pairs. Let p_i and p_j be consecutive hubs used in OPT_1^* such that $x_i \leq x_j$ but $x'_i > x'_j$. We will show that swapping p_i and p_j's final locations does not increase the budget, i.e. if $b = |x_i - x'_i| + |x_j - x'_j|$ and $b' = |x_i - x'_j| + |x_j - x'_i|$ then $b \geq b'$ holds. On the other hand, the number of unordered pairs decreases by one. This contradicts the minimality of the number of unordered pairs. Note that, since we only swap the final locations of the hubs, the connectivity is preserved. Further, swapping does not change the total budget used in the y direction. Therefore, the overall budget does not increase as well.

Assume that we fix the locations of x_i and x_j: we have three "bins" ($x \leq x_i$, $x_i < x \leq x_j$ and $x_j < x$) for possible locations of x'_i and x'_j. The following set of equations correspond to all 6 possible cases. In each case, the claim above holds. In Figure 3.4, the second statement in the first line is illustrated.

$$
\begin{aligned}
x'_j < x'_i \leq x_i \leq x_j &\Rightarrow b = b' \\
x'_j \leq x_i < x'_i \leq x_j &\Rightarrow b > b' \\
x'_j \leq x_i \leq x_j < x'_i &\Rightarrow b \geq b' \\
x_i \leq x'_j < x'_i \leq x_j &\Rightarrow b > b' \\
x_i \leq x'_j \leq x_j < x'_i &\Rightarrow b \geq b' \\
x_i \leq x_j \leq x'_j < x'_i &\Rightarrow b = b'
\end{aligned}
$$

□

We now solve *SumDist* optimally for the L_1 metric (up to an arbitrarily small additive cost). We start by building a table $T(k, i, B)$ which stores the maximum *reach* using k hubs subject to: (i) the i^{th} robot is the k^{th} hub, and (ii) the budget for the first k robots is at most B. The entries are computed as follows:

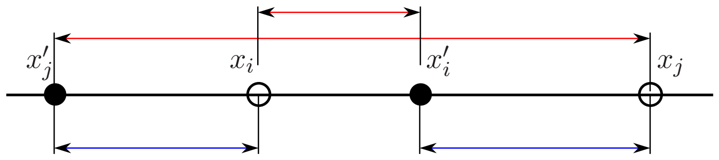


Figure 3.4: Figure shows the case: $x'_j \leq x_i < x'_i \leq x_j$. Upper line segments show the total cost for the initial solution and lower line segments show the costs after swapping. When we swap the final locations of robots, we decrease the total cost while satisfying the ordering property.

$$T(0,i,B) = r \ \ \forall_i \tag{3.4}$$

$$T(k,i,B) = 0 \ \ \forall_{k>i} \tag{3.5}$$

$$T(k,i,0) = \begin{cases} x_i + r & \text{if a } k \text{ hub } \textit{bridge} \text{ exists initially} \\ 0 & \text{o/w} \end{cases} \tag{3.6}$$

$$T(k+1,i,B) = \max_{k \leq j < i} \max_{b' \in C(x_i)} \min(T(k,j,B-b), x_i + b') + r \tag{3.7}$$

$$T(k,i,B+\varepsilon) = \max_{k \leq j < i} \max_{b' \in C(x_i)} \min(T(k,j,B+\varepsilon-b), x_i + b') + r \tag{3.8}$$

where B is discretized by ε, $b' = b - y_i$ and $C(x_i)$ is a set of possible values for b'. We will discuss ε and $C(x_i)$ shortly. The first two equations are the base cases. If initially the robots create a communication bridge between s and p_i with k hubs, then Equation 3.6 sets the *reach* $T(k,i,0)$ to $x_i + r$. This can be checked by building a graph G whose vertices are $P' \cup \{s,t\}$ where P' is the set of hubs that are initially on $[s,t]$. There is an edge between two vertices if the distance between them is at most r. If G has a path between s and p_i of length at most k, then a communication bridge from s to p_i can be formed with budget 0.

Here, we discuss only how to extend the first dimension of the dynamic programming formulation (Equation 3.7). The argument for the other dimension (Equation 3.8) is similar.

To calculate $T(k+1,i,B)$, we consider the optimal *reach* with k hubs when using the p_j as the k^{th} hub for all $j < i$ (due to the ordering property we do not need to consider the locations of earlier hubs in the optimal solution). Let $R = T(k, j, B - b)$ be the maximum *reach* achievable by using k robots with p_j as the last hub and a total budget of $B - b$. The final location of p_j in this optimal *reach* is $R - r$. We need to compute the *reach* for $k+1$ hubs where p_i is the last hub and p_i travels at most b units. For this, we consider all possibilities for R.

Note that the distance of the initial location $u_i = (x_i, y_i)$ to L is y_i. Hence, $b \geq y_i$ must hold for p_i to act as a hub. Let $b' = b - y_i$, then, $[x_i - b', x_i + b']$ is the region that robot p_i can be placed on the line L with a budget of b'. Due to the ordering property, p_i must be placed to the right of p_j. Therefore, its location is after $R - r$ and before R (otherwise p_j and p_i cannot communicate). In other words, valid locations for p_i are given by the intersection of $[x_i - b', x_i + b']$ and $[R - r, R]$, and this set should be non-empty.

We now compute the set of valid budgets b for robot p_i. Since the robot has to travel y_i for the vertical component, the remaining budget for the horizontal component is $b' = b - y_i$. Let $C(x_i)$ be the set of possible values for b'. This set is computed as follows:

$$C(x_i) = \{b' | b' \leq B - y_i \wedge Z(x_i, R)\} \tag{3.9}$$

$$Z(x_i, R) = \begin{cases} R - r - x_i \leq b' \leq R - x_i & \text{if } x_i \leq R - r \\ 0 \leq b' \leq R - x_i & \text{if } R - r < x_i \leq R \\ b' = x_i - R & \text{o/w} \end{cases} \tag{3.10}$$

For a budget b' to be valid, we must have $b \leq B$. This gives the first condition for b': $b' \leq B - y_i$. We use the function Z to constrain b' as a function of x_i and the current *reach* R. We consider the three cases based on the location of x_i with respect to the location of the last robot (x'_j) and the *reach* $R = x'_j + r$. See Figure 3.5.

Case 1 ($x_i \leq x'_j$): In this case, we must have $b' \geq R - r - x_i$, (otherwise p_i cannot extend the current *reach*) and $b' \leq R - x_i$ (if p_i moves further to the right, p_j and p_i can't communicate).

Case 2 ($x'_j < x_i \leq R$): Similar to case 1, b' should not be greater than $R - x_i$. The lower bound is obtained by the non-negativity of b'.

Case 3: When x_i is to the right of the current *reach* R, there is only one value robot p_i should move: the rightmost reachable point.

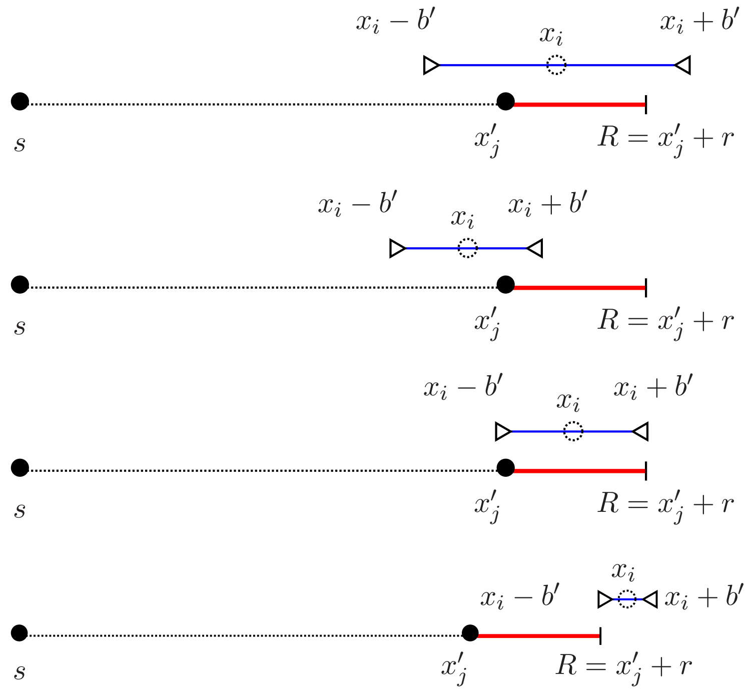


Figure 3.5: Let x'_j be the last hub location at the *reach* and p_i be the robot considered at the current iteration. **Top Figure:** When b' is too large both end points of feasible region is out of the the region $[x'_j, R]$, hence b' is redundant in this example. **Next Three Figures:** The three cases considered in Equation 3.10 are illustrated.

The new *reach* after placing robot p_i to $min(R, x_i + b)$ is $min(R, x_i + b) + r$. In order to compute $T(k+1, i, B)$, among all possible $j < i$ and all possible budgets $b' \in C(x_i)$, we find the optimal *reach*. Since the size of the range of possible values of b' is bounded by r, the size of the set $C(x_i)$ is at most r/ε. Hence, each entry can be calculated in $O(nr/\varepsilon)$ time.

We now show how this result yields an approximation algorithm for L_2. Let OPT_1^* and OPT_2^* be optimal solutions for L_1 and L_2 metrics, respectively. The following lemma bounds the deviation between OPT_1^* and OPT_2^*.

Lemma 15. *Let OPT_2^* be the optimal solution for the L_2 metric with a given budget B. Suppose OPT_2^* can connect s and t using k hubs. There exists optimal solution OPT_1^**

for the L_1 metric which can connect s and t by using k hubs and a budget of $\sqrt{2}B$.

Proof. Let (x_i, y_i) be the initial location of a robot used in OPT_2^* and x_i' be the final location. The L_1 and L_2 distances are $|x_i - x_i'| + y_i$ and $\sqrt{|x_i - x_i'|^2 + y_i^2}$, respectively. Without loss of generality, we scale the distances by $1/y_i$ so that the L_1 and L_2 distances become $a+1$ and $\sqrt{a^2+1}$, respectively where $a = |x_i - x_i'|/y_i$. From elementary calculus, it is easy to show that: $f(a) = \frac{a+1}{\sqrt{a^2+1}} \leq \sqrt{2}$. □

To obtain the optimal L_1 solution for budget B, we solve $T(k, i, B)$ for all possible k, i, B (where B is discretized with ε intervals). Due to the discretization, the total budget used here can be at most $k_1^*\varepsilon$ than the budget used by OPT_1^* where k_1^* is the number of hubs used by OPT_1^*. In other words, our dynamic programming algorithm can find a solution with k_1^* hubs by using at most $B_1 + k_1^*\varepsilon$ budget where B_1 is the used budget with L_1 metric. This means that B', the total budget used by our solution will be bounded by $B_1 + n\varepsilon$. Consequently, the total budget used by our algorithm will be at most $\sqrt{2}B + n\varepsilon$ where B is the given budget in L_2 metric. We can choose ε to achieve an arbitrarily small additive error.

We now establish the running time of the algorithm. The size of the table is $O(\frac{n^2B}{\varepsilon})$ and as we discussed earlier each entry can be calculated in $O(nr/\varepsilon)$ time. Hence, the time complexity of our algorithm is $O(\frac{n^3Br}{\varepsilon^2})$.

Theorem 16. *If there exists a solution to SumDist that uses k hubs such that the total movement of robots is B in the L_2 (Euclidean) metric, then we can compute a solution where we use at most k hubs and the total movement of robots is at most $\sqrt{2}B + n\varepsilon$ in $O(\frac{n^3Br}{\varepsilon^2})$ time, where ε is the discretization constant.*

3.3 Bounds on Number of Hubs

Let $OPT(d)$ be the number of hubs in an optimal solution to *MaxDist* with distance constraint d. How does this constraint affect the number of hubs on the bridge? In other words, if $OPT(\infty) = \lceil |st|/r \rceil - 1$ is the number of hubs required in the unrestricted version, how far is $OPT(d)$ from $OPT(\infty)$? In this section, we show that $OPT(d)/OPT(\infty) \leq 2$.

Assume that $m-1 < |st|/r \leq m$, for some integer $m > 1$. (The case $m = 1$ is uninteresting, as s and t are then within distance r, hence connected.)

Partition $[s,t]$ into m equal-length intervals, labeled from s to t as $I_1, I_2, \ldots, I_m$. Each interval has length greater than $(1-1/m)r$ and at most r. Consider any solution for $OPT(d)$. This solution connects s and t with the fewest number of hubs. In such a solution, we can have at most two hubs inside any I_j, $2 \leq j \leq m-1$. To see this, note that if there were three or more hubs in I_j, then all but the two extreme ones in I_j could be removed without losing connectivity (since the length of I_j is at most r), thereby obtaining a solution for $OPT(d)$ that has fewer hubs than the original optimal solution—a contradiction. Along similar lines note that I_1 and I_m can each contain at most one hub; if there was more than one hub in I_1 (resp. I_m), then all the ones except the one farthest from s (resp. t) can be removed without losing connectivity.

It follows that for any optimal solution, we have $OPT(d) \leq 2(m-2)+2 = 2(m-1)$. Also, $OPT(\infty) = \lceil |st|/r \rceil - 1 = m-1$. Hence, we have the following.

Lemma 17. *$OPT(d)/OPT(\infty) \leq 2$.*

Using similar arguments, it can be shown that the same bound applies for *SumDist*. We omit the details.

Next, we show that the bound in Lemma 17 is tight: We claim that, for any finite d, there is an instance of *MaxDist* with the optimal solution $OPT(d)$ for which $OPT(d)/OPT(\infty) = 2$.

Let $|st|/r = m > 1$; thus, each interval $I_1, I_2, \ldots, I_m$ defined above has length r. Let ε be a real number in the (open) interval $(0, \frac{r}{m-1})$. Consider a set $V = \{v_1, v_2, \ldots, v_{2(m-1)}\}$ of points on $[s,t]$, defined as follows: for $j = 2, 4, \ldots, 2(m-1)$, $v_j = \frac{j}{2}\varepsilon + \frac{j}{2}r$, and for $j = 1, 3, \ldots, 2m-3$, $v_j = \frac{j+1}{2}\varepsilon + \frac{j-1}{2}r$. See Figure 3.6.

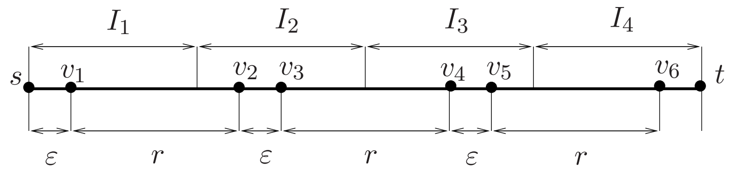


Figure 3.6: Selection of points $v_1, v_2, \ldots, v_{2(m-1)}$ on $[s,t]$, with $m = 4$.

The set V satisfies the following (easily-verifiable) properties: (i) $v_1 \neq s \in I_1$ and $v_{2(m-1)} \neq t \in I_m$; (ii) successive points in $V \cup \{s,t\}$ are within distance r; and (iii) at least one pair of successive points in $V' \cup \{s,t\}$ is not within distance r for any $V' \subset V$.

Let $\mathcal{P}$ be a set of $n \geq 2(m-1)$ robots $\{p_1, p_2, \ldots, p_n\}$ and choose their initial positions in $\mathbb{R}^2$ as follows: for $j = 1, 2, \ldots, 2(m-1)$, place p_j at initial position $u_j = (v_j, d)$. Place any remaining hubs in $\mathcal{P}$ at some distance greater than d from $[s,t]$.

Observe that only $p_1, p_2, \ldots, p_{2(m-1)}$ can move onto $[s,t]$ and, moreover, each such p_j can move only to the location v_j. By properties (ii) and (iii) above, it follows that $p_1, p_2, \ldots, p_{2(m-1)}$ are necessary and sufficient to establish a communication bridge between s and t. Therefore, $OPT(d) = 2(m-1)$ and the claim follows.

3.4 Concluding Remarks

In this chapter, we introduced the problem of building a communication bridge between two points s and t while minimizing the number of hubs on the bridge and satisfying a maximum (or total) distance constraint for the robots. For both versions we presented constant factor approximation algorithms for the geometric version where the robots must move onto $[s,t]$.

There are many interesting directions for future work. It is not clear whether the $\sqrt{2}$ approximation factor for the geometric version can be improved. The general version in which the final locations of hubs can be anywhere on the plane seems difficult. Solving the version where there are multiple source and destination pairs seems to be even harder.

Acknowledgment

The results of this chapter appeared in ALGOSENSORS 2009 [82]. This work is joint with Yokesh Kumar, Prof. Volkan Isler and Prof. Ravi Janardan.

Chapter 4

Data Mules

A Wireless Sensor Network (WSN) is a network of wireless sensing devices which is deployed over an area of interest to collect environmental data. Each sensor in a WSN consists of a low-power CPU, a small data storage unit, a wireless radio with limited communication range, and a set of sensors to measure physical phenomena of interest such as motion, temperature, humidity, etc. These embedded devices are mostly powered by limited energy sources such as a pair of AA batteries. Figure 4.1 shows a common sensor called *mote* which is extensively used by WSN researchers. Recent advances in WSN technology have reduced the costs of these sensors to about \$100 per sensor. With this progress, now it is feasible to use WSNs in environmental monitoring tasks. In *environmental monitoring*, scientists collect statistical data (e.g. temperature) from the environment which has to be deployed for years over large geographic areas [17]. Although, reduced cost makes a WSN practical for environmental monitoring, gathering the collected data efficiently from the sensors stills remains a big challenge.

In the literature, there are two primary techniques for gathering the collected data from sensors. The first method is to manually gather the collected data [83, 84]. This process requires a person to visit each sensor in the environment and manually download the data from the sensor. However, since these sensors are sparsely deployed over large areas and they monitor the environment for long period of times, this process requires extensive labor.

In the second method, a connected WSN is deployed to transfer the collected data over the network. Since the wireless range of sensors is limited (50-100 meters), these

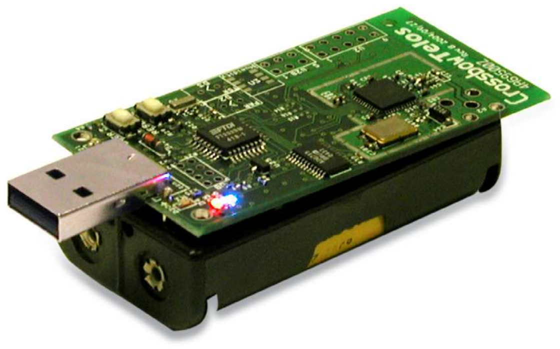


Figure 4.1: TelosB mote is an embedded sensing platform developed at UC Berkeley. TelosB consists of a USB interface, a micro-controller unit with extended memory, IEEE 802.15.4 radio with integrated antenna and optional build-in sensors. It is operated by an open-source operating system called TinyOS and powered by two AA batteries [2].

networks must be deployed dense enough to ensure network connectivity. On the other hand, in most environmental monitoring applications, the environment is very large and the sampling locations are far apart from each other. Hence, a large number of sensors is required just to relay messages which makes this method very costly if not infeasible. As an example, deploying a uniform grid of resolution 100 meters over an area of 5 km by 5 km might cost as much as $300,000.

The last decade has witnessed tremendous advancements in mobile robot and autonomous navigation technologies. It is now feasible to build low-cost and robust outdoor mobile robots that can accomplish complex navigation tasks. Therefore, it is possible to use robots as *data mules* where robots periodically visit sensor nodes, gather the collected data from sensors, and carry it back to a gateway or a server where the data is aggregated. Note that there is a delay between the time the data is collected and the time it reaches the gateway. However, this delay is not significant in an environmental monitoring application because the aim is to collect statistical information about the environment over long period of times. Figure 4.2 shows an illustration of the proposed data mule system. This alternative method provides cost-effective and energy-efficient solutions for the data gathering task. Next, we discuss these advantages of data mules compared to a dense WSN network deployment.

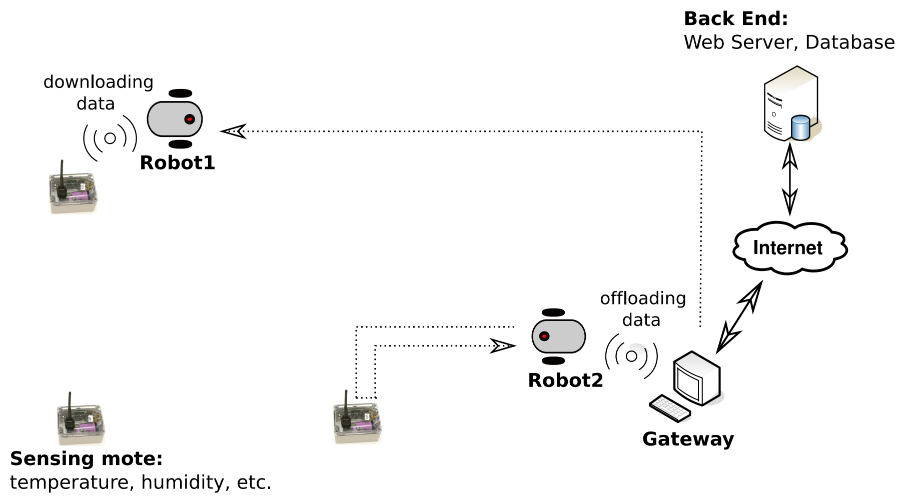


Figure 4.2: Overall system architecture, including a number of sensing motes, multiple robots acting as data mules, and a gateway to which robots offload collected data and receive further commands. Each robot communicates with the sensing nodes and the gateway through a locally connected mote. All collected data is eventually stored in a back-end database for further processing and visualization.

In Section 4.5, we present the design details of an outdoor robot platform. This robot was built in our lab and its cost is on the order of $1000. This shows that data mules could drastically reduce the costs of environmental monitoring applications. In addition, the deployment and maintenance costs for data mules are minimal. Dense networks require extensive labor due to the deployment of a large number of sensors. On the other hand, only a small number of sensors (sensing nodes) has to be deployed and maintained in a data mule system, which further cuts the labor costs.

Since motes are mostly battery operated, limited energy is another key challenge for WSNs. The mote's wireless radio is its largest energy consumer. These devices operate at 1 milliwatt while running with low frequency. On the other hand, most radios consume about 20 milliwatts when on. This observation means that the motes can spend the majority of their energy stores transmitting collected data, especially because they may need to forward data for other motes in the network. Moreover, some nodes in the network might act as sink points which transfer data forwarded from multiple nodes. This leads such nodes to die early on and as a consequence the network could suffer from broken communication links.

Robots are capable of carrying large batteries and can recharge themselves autonomously or optionally they can carry a solar panel to recharge themselves on the go. Hence, robots do not suffer from limited energy sources. Rather, we can use robots' mobility to extend the life-time of the sensor network. First of all, since robots directly download the data from sensors, collected data has to be transmitted just once. On the other hand, in a connected network, data has to be transmitted multiple times over the network until it reaches the gateway. In addition, since robots can get close to the sensors, the communication range of the transmitters on the sensors can be reduced, which leads to further energy savings. In Section 4.3, we justify these claims with empirical results by comparing a proof-of-concept data mule system to a grid WSN deployment. In Section 4.4, we improve the life-time of the network by searching efficiently for a good location before downloading the data. Since the robot's energy is not the bottleneck in the system, we can use the robot to find a location where the packet loss rate is minimal. Hence, we can reduce the retransmission of packets, which further improves the energy savings.

A crucial problem that arises in data muling applications is the problem of planning the routes of robotic mules: given locations of n sensors, compute the routes of k robots so that the time to download the data from all sensors is minimized. We call this problem the *Data Gathering Problem (DGP).* A problem similar to DGP is the well-known *Traveling Salesperson Problem (TSP)* which asks for the shortest path for a salesperson to visit n cities (sensor locations) [85]. There is also a variant of TSP, called k-TSP, where k travelers visit n cities, and the objective is to minimize the length of the maximum tour [86]. Unlike the k-TSP problem, robots do not have to go all the way to the sensor locations in DGP. Instead a robot can download a sensor's data when it is inside the sensor's communication range. Moreover, visiting each sensor incurs a cost in terms of the download time of the data which is not addressed by k-TSP. Hence, a solution to the DGP problem not only depends on the sensor locations as in k-TSP but also depends on the communication model of the sensors.

In our previous work [25], we addressed the DGP problem where the communication range of the sensors is modelled as a disk of radius r. The download time is defined to be T and identical inside the disk. A similar problem formulation to the disk model is called TSP with Neighborhoods (TSPN) where n regions (e.g. disks) are given and the goal is to find the shortest path that visits each neighborhood [3]. Even though DGP under disk communication model resembles TSPN, there are important differences.

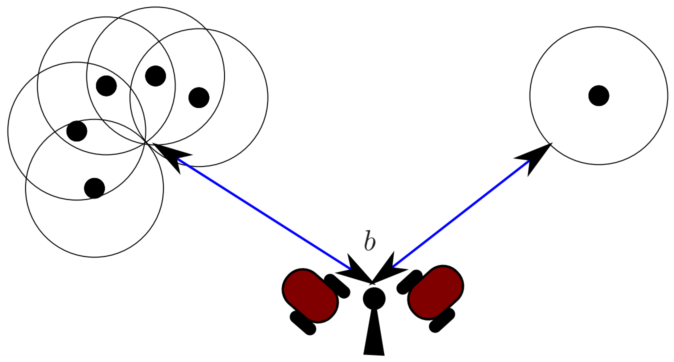


Figure 4.3: Two robots charged with collecting data from the sensors and relaying them to the base station b. The filled circles correspond to sensor locations. The circle around a sensor illustrates its communication range. The figure shows optimum TSPN tours for the two robots that minimize the maximum distance traveled by any robot. This solution is not appropriate for data gathering because the robot assigned to the left group would spend significantly more time downloading the data from the sensors.

For example, consider the scenario illustrated in Figure 4.3 where the optimal TSPN tours for two robots are shown. Note that the objective of TSPN is to minimize the travel distance. However, in data gathering, downloading the data takes time. If we use the TSPN solution, the robot on the left can spend significantly more time to download the data from all assigned sensors. In fact, we can make the TSPN solution arbitrarily bad by increasing the number of nodes in the left cluster. We present a literature on TSP and its variants in Section 4.2.3.

The disk communication model is a simplistic model which ignores the fact that the radio signal propagation is affected by environmental factors. As an example, a radio signal might follow multiple paths possibly due to signal reflection from obstacles such as buildings, rocks, etc. Hence, a receiver might receive signals through multiple paths from the same signal source. When these signals interfere, the signal strength might increase or the signals might cancel each other causing a weak signal. This phenomenon is called *multi-path effects* of the radio signals. In Section 4.5, we modify the DGP algorithm by utilizing the multi-path effects. In this modified version, the robot opportunistically downloads from a sensor that it receives a good signal from even though it is not in the communication disk of the sensor. We show the practical improvements of this modification through field experiments.

Recent research [26] and our previous experimental results suggest that sensor's communication can be modelled better using two concentric disks. In this model, if the robot is a distance r_{in} away from sensor s, it can download the sensor's data in T_{in} units of time. If the distance is greater than r_{in} but less than r_{out}, the download time is $T_{out} > T_{in}$. Otherwise, the robot can not download the data from s. We refer to this model as the *two ring model*, and the problem of downloading data from a given set of sensors in the minimum amount of time under this model as the *Two-Ring Tour (TRT)* problem. In Section 4.6, we present routing algorithms for this model and demonstrate its effectiveness through field experiments.

We start the chapter with the literature survey on data mule systems in Section 4.1. We present background material on WSNs and TSP related problems in Section 4.2. In Section 4.3, we present a proof-of-concept data mule implementation. In this section, we compare the energy consumption using data mules over using a connected network. In the Section 4.4, we discuss how we can further improve energy savings by using robots to find a good location prior to start downloading data. In Section 4.5, we present an opportunistic DGP algorithm. We present the details of an inexpensive outdoor robot platform and the field experiments conducted with this robot. In Section 4.6, we present routing algorithms under the two ring communication model and field experiments to show the practical feasibility of this model.

4.1 Related Work

In recent years, using mobility for carrying data over networks has received significant attention. In a Mobile Ad-hoc Network (MANET), each device is mobile and can move independently from the others. Since the communication links are constantly changing, the main challenge in a MANET is to route the information from one node to another. Various routing protocols [32, 35–37] are proposed in the literature to reliably transfer data over MANET. Although, most of the existing literature focuses on such settings where the mobility of the network entities is not controlled, recently, scientists have started to utilize controlled mobility to overcome some challenging networking tasks. As an example, robots are now used to efficiently gather data from sensors deployed over large environments. In this framework, robots act as data mules to get close vicinity with sensors, download sensor data and carry the collected data back to a base station. In this section, we present an overview of the related work on data mules.

In [18], the authors present a data muling system which uses uncontrolled entities (such as animals, humans with wireless devices) for carrying data. The authors present a three tier sensor network architecture. The bottom layer consists of a sparse sensor network. In the middle layer there are mobile entities such as vehicles and humans which carry the data from bottom layer to the access points in the top level. These ideas were also implemented in real systems in ZebraNet [87] and Smart-tag [88] projects.

In the next set of results, the mobile agent is still uncontrolled however the mobility model knowledge is used to optimize the various parameters of the network. In [89], the authors explore the use of observed mobile agents in reducing the energy consumption in sensor networks. In [90], data mules' trajectories are parallel line segments and the goal is to find the balanced assignment of sensors so that the tour times of data mules is minimized. In [91], the authors explore the transmission scheduling i.e. schedule to wake up a node and transmit to the data mule. The solution is presented under the assumption that the mule follows a random-waypoint mobility pattern. An adaptive data transfer protocol which minimizes the time interval for a single sensor to offload the data is presented in [92]. This protocol considers the packet loss rate due to the distance between sensor and the data mule at the time of offload. A simple protocol is presented in [93] in which nodes send periodic beacons with low duty cycle and turns on their radio when the connection with a data mule is satisfied. A recent review on the state of the art in exploiting sink mobility can be found in [94–97]

The study of systems which use controllable agents as data mules is relatively recent. A variant of the Data Gathering Problem (DGP) was considered as a scheduling problem in [24]. Kansal et al. consider the control of the robot's velocity along a fixed path to improve transmission quality. They do not address the computation of the path. In their subsequent work, Kansal et al. proposed a coverage trajectory to collect data from sensors which are uniformly deployed in a circular arena [98]. Similarly [99] study the speed control problem for the data mule when it is downloading data from a sensor while traveling within its communication range. Authors present a heuristic which computes the speed change along the mule's path and a schedule for downloading the data from the sensors. DGP is also considered as a coverage problem. In [100], the authors aim to find a path for each robot such that all paths are disjoint, all sensors are covered and all paths are of equal length. They investigate the performance of their algorithms in simulation. In [101], the authors present a heuristic for the multi-robot boundary coverage problem. In a boundary coverage problem we are given k robots and n convex, two-dimensional objects. The goal is to find a tour for each robot such that all points on the boundary of each object are inspected and the inspection load is balanced. None of these algorithms present any theoretical guarantees on the performance of the paths. Several other mobility patterns are proposed in the literature where the life-time of a stationary network with stationary base station is compared over a stationary network with a mobile base (mobile sink) [102–104].

In the following body of work, variants of DGP are formulated as TSP instances. In [105], the authors present heuristics to schedule visits of a mobile agent to collect data from cluster heads. The heuristics focus on data latency and data aggregation rate of clusters. In our work we focus on the travel time of the mobile agents and present algorithms with theoretical performance guarantees. [106] present a heuristic approach for minimizing path length in data muling. Authors consider spatially separated WSNs which are to be connected by a data mule. The objective is to find a path for mule which visits one sensor each in each WSN. Their heuristic creates a path from the convex hull of the set of sensors, choosing one sensor from each WSN, and modifies the path to add the sensors from WSNs not covered by the convex hull. In [107], the authors formulate the problem of collecting sensor data using a single robot as an instance of the TSP with neighborhoods (TSPN) problem. They do not address the time spent in downloading data. As shown in Figure 4.3, ignoring transmission time can worsen the performance of the system drastically. None of the work that uses TSP-based heuristics has been implemented and tested on a real system.

Implementations of data mule system are presented in [108]. In [108], the authors present an underwater data muling system. In the underwater scenario, sensors and underwater vehicles communicate through optical communication, which requires a close proximity as well as a good view-angle to start communication. The authors use a TSP algorithm to compute the trajectory of the robot. However since GPS localization is not available under the water, the vehicle has to navigate under high uncertainty and periodically surface to get GPS fix. This makes the designing global routing algorithms challenging. The authors propose spiral movement for the robot to find the sensors. This strategy is not efficient for our scenario, in which the sensor locations are known and the robot can localize itself.

4.2 Background

In this section, we present background information on WSNs and the communication protocol between sensors and robots. We also present the detailed background and literature search on TSP and its variants since we use similar techniques throughout the chapter.

4.2.1 Wireless Sensor Networks

A wireless sensor network (WSN) is a collection of small embedded computing devices (motes) deployed over an area of interest, communicating through wireless radios. These motes consist of a micro-controller, data storage devices, sensors, analog-to-digital converters (ADCs), a data transceiver, and an energy source (e.g., AA batteries). Existing motes use an 8 or 16-bit micro-controller with tens of KBytes of RAM, hundreds of KBytes of ROM for program storage and external storage in the form of FLASH memory [109]. These devices operate at one milliwatt while running at about 10 MHz. Most of the circuits can be powered off, so the standby power can be about one microwatt. If such a device is active 1 percent of the time, its average power consumption is just a few microwatts enabling long term operation even with two AA batteries. Mote radios transmit at rates between 10-250 Kbps, consume about 20 milliwatts when in transmit, receive, or idle listening mode, while their range typically is measured in tens of meters.

One of the most popular wireless sensor network applications is environmental monitoring [17,110]. In its basic form, environmental monitoring involves reliably gathering the measurements from each of the network's motes. Furthermore, in order to observe long term spatial and temporal trends, these networks need to be deployed for years and cover large geographic areas. These requirements mean that such networks must have duty cycles of 1% or lower and be deployed sparsely over the monitored area.

4.2.2 Sensor and Robot Communication

As previously mentioned, sensing motes keep their radios mostly off to conserve energy. Then to check for any pending requests, each sensing mote transmits a beacon and waits for an acknowledgment. If no acknowledgment arrives the mote turns its radio off, otherwise it remains active waiting for data download requests. This *Low Power Probing* (LPP) mechanism was initially proposed in [111]. By controlling the beacon frequency the node can control its duty cycle. For example, selecting a beaconing interval of 20 seconds, leads to a $\sim 0.1\%$ duty cycle, as a beacon-acknowledgement cycle lasts 20 msec.

Based on this description, the robot moves close to the sensing mote, waits for a beacon message, acknowledges it, and then issues a command to the mote to download (some of) its data. An alternative wakeup strategy is to use a lower power sensor to wake up the node. For example one could use the mote's light sensors or even a Reed switch (i.e. an electrical switch operated by an applied magnetic field) to wake up the mote. In this case, the robot could shine a light or carry a magnet that would activate the sensor which will in turn activate the sensing mote's radio for the subsequent download.

Once the mote is awake, the robot transmits a request containing the range of collected data it needs to download from the mote. The actual data download process uses the standard NACK-based Automatic Repeat reQuest (ARQ) protocol.

4.2.3 TSP and Variants

The Traveling Salesman Problem (TSP) is one of the most studied combinatorial optimization problems. Even though TSP in its general form is inapproximable, the metric version admits a constant factor approximation [112]. The Euclidean version in any fixed dimension admits a Polynomial Time Approximation Scheme (PTAS) [113, 114]. A generalization of TSP which received significant recent attention is TSP with Neighborhoods (TSPN) [3, 115, 116]. In a TSPN instance, instead of n points we are given n neighborhoods (a neighborhood is a bounded region). The goal is to find a minimum cost tour which visits at least one point in each neighborhood.

Arkin and Hassin [115] introduced TSPN and presented a $(3\sqrt{2}+1)p$ approximation algorithm for the case when neighborhoods are equal length parallel segments (Here, p is approximation ratio for TSP). For neighborhoods which are translates of a convex region, they gave a $(\sqrt{7^2+3^2}+1)p$ approximation algorithm.

Elbassioni et al. presented a constant factor approximation algorithm for intersecting fat convex objects of comparable diameters where the tour is restricted to visit each object only at a finite set of specified points [116].

The version of TSPN where the neighborhoods are uniform disks has many applications. In fact, TRT is a generalization of this problem which in turn generalizes Euclidean TSP. Mitchell et al. presented constant factor approximation algorithms for TSPN with uniform disks [3]. In this work, they start with the case where the disks are disjoint. They show that a TSP tour that visits the centers of the disks is an $O(1)$-approximation to the TSPN tour. Mitchell et al. also showed that when the disks are disjoint, TSPN admits a PTAS. The result generalizes to other "fat" regions on the plane [113]. The algorithm can be modified to yield a PTAS for k-TSPN algorithm where the goal is to find the shortest tour which visits at least k disks out of n disks in the input. k-TSPN with uniform (intersecting) disks also admits a PTAS [117].

4.3 A Proof-of-Concept Design

Now, we present a proof of concept system we developed, as well as results from a small experiment that proves the feasibility of using mobile robots as data mules and evaluates the energy savings achieved by using robots.

Our proof of concept consists of a network of sensing motes and a number of mobile robots. Each sensing mote is a Tmote Sky from Moteiv [109]. Moreover, each of the Acroname Garcia robots has a Stargate PDA-class single board computer, that controls the robots' motors and sensors (see Figure 4.4). The Stargate uses a Tmote Sky motes, directly connected to its USB interface, to communicate with the sensing nodes. The communication between the mote and the robot's computer follows a simple command-and-response protocol that we developed. Once the robot approaches a sensing node, it waits for a beacon message to verify that the node is still alive. After it receives the beacon, the robot initiates the data download process. If on the other hand, a beacon is not received within a predetermined amount of time, the robot proceeds to the next sensing mote on its path.

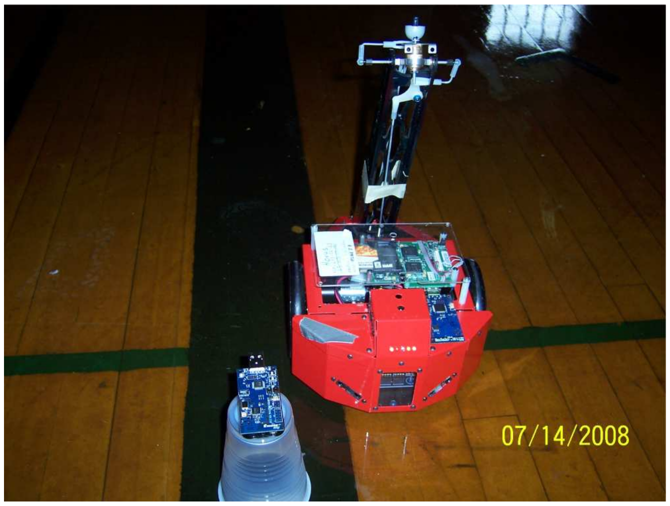


Figure 4.4: An Acroname Garcia robot visiting a sensing mote. Note the single board computer on the robot under the clear lid and the mote connected to the robot's computer.

We assume that the motes' transmission power is set to its lowest level to reduce energy consumption. Hence, while planning the mules' trajectories, we set the visiting locations to be the motes' exact locations. The trajectories of the multi-robot system are pre-calculated as k-TSP tours. Figure 4.5 illustrates the resulting k-TSP tours for $k = \{1, 2, 3\}$ on a regular grid where twelve motes are deployed. For this small TSP instance, we find the optimal TSP tour by enumeration, which is feasible for this rather small network. For larger problems, efficient TSP solvers can be used [118]. To compute the 2-TSP and 3-TSP tours, we split the optimal TSP tour into smaller tours using the k-SPLITOUR algorithm [86].

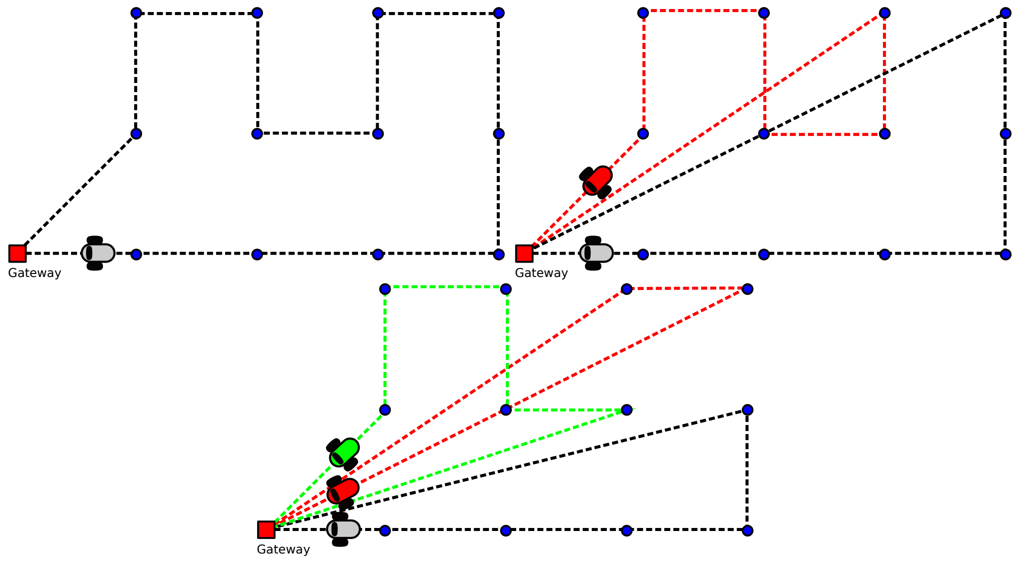
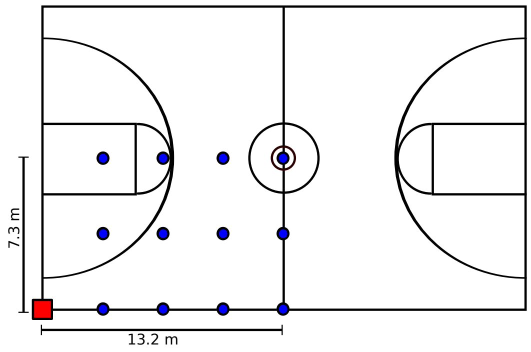


Figure 4.5: Optimal TSP tours for one, two, and three data mules. The red square represents the gateway and blue circles represent mote locations.

We illustrate the practical feasibility of data gathering using data mules with a small-scale experiment. For this experiment, we deployed twelve motes and a gateway on a virtual grid on a basketball field at the Rensselaer Polytechnic Institute (see Figure 4.6(a)). The robot system we developed for the experiment follows a step by step procedure. In each step, a robot takes a turn or takes a forward step with a length of at most 7 m. We limit the length of each step to limit the robot's deviation from its designed path. Such deviations occur due to the inherent accuracy limits of the robot's motors. Because the goal of this experiment is not to evaluate the accuracy of indoor localization techniques and autonomous repositioning, we manually re-position the robots to correct the position if they veer off the desired path. In total, we conducted three experiments with one to three data mules. Figure 4.5 shows the robots' trajectories for each of the three experiments and Figure 4.6(b) shows a snapshot from the experiment with two data mules. Link to the videos of all experiments can be found in Appendix A-Multi Media Extension No: 2.



(a) Map of the deployment area. The deployment consists of twelve motes (blue circles) and a gateway (red square), placed on a grid that spans a quarter of a basketball court.

(b) A snapshot from the experiment with two mules. One mule (white robot) is following its designed path while the other mule (red robot) is downloading data from the assigned mote.

Figure 4.6: The map of our deployment (left) and a snapshot from the experiment with two mules (right).

4.3.1 Evaluation

As described at the beginning of the chapter, one key reason for using data mules is to reduce the sensing motes' energy consumption and thus prolong the network's lifetime, while guaranteeing reliable data collection. We quantify this benefit by comparing the energy consumed to send a single packet from each mote in two settings. In the first setting, an end-to-end, multi-hop wireless network system is used to collect the data. In the second setting, sensing motes directly offload their packets to data mules. We simulate this comparison based on a grid of twelve motes whose pattern mirrors the 3×4 grid that we deployed in the basketball court. The deployment is shown in Figure 4.6(a).

On top of this, we make the following assumptions. First, we assume that each mote can communicate only with its direct neighbors on the grid and therefore the longest end-to-end network path spans four wireless links. Second, each packet can be lost with probability p, which is uniform across all of the network's links. Finally, we use a hop-by-hop retransmission scheme in which case each node will retransmit the packet if it does not receive an acknowledgment from its upstream neighbor.

Given these assumptions, we calculate the sum of the expected number of transmissions (ETX) necessary to have each of the network's twelve motes successfully deliver a single packet to the gateway. Based on the end-to-end routing tree overlayed on the mote grid, the aggregate ETX is equal to 30 when $p = 0$ (i.e., no packet loss). On the other hand, when data mules are used, the aggregate ETX is equal to 12 since each mote can directly offload its packet to a mule over a short and reliable wireless link. While the benefits of using data mules are clear even with error-free radio links, as Figure 4.7 suggests, the disparity rapidly increases as p increases. Furthermore, this estimate assumes that the gateway can directly communicate with some of the network's motes. In reality, however, the gateway may not be in sensing motes' communication range in which case one or more *relay points* are necessary. Adding relay points, however, increases the length of the end-to-end path and thus further increases the ETX.

The second benefit of using data mules is that motes can decrease their transmission power and thus save energy. Figure 4.8 supports this claim, showing that if a mote reduces its transmission power level from its default level of 1 mW (0 dBm) to 0.003 mW (-25 dBm), the energy consumption is reduced drastically. Furthermore, because the mules can reliably approach sensing motes to a close distance, reducing the transmission power does not increase the probability of loss.

In addition to estimating the energy the sensing motes consume, we also consider the robot's energy consumption. Since robot batteries are also limited, yet rechargeable energy sources, we need to consider how long the batteries last when we dispatch the robots to collect data. If the network is too big and therefore the tour is too long, the energy stored in a single robot battery may not be sufficient to complete a single tour. In that case, we can split the the single tour until we get feasible tours that each robot can complete at least once, before recharging its battery. Even if a single robot can tour the whole network, we need to be able to predict how many tours it can complete

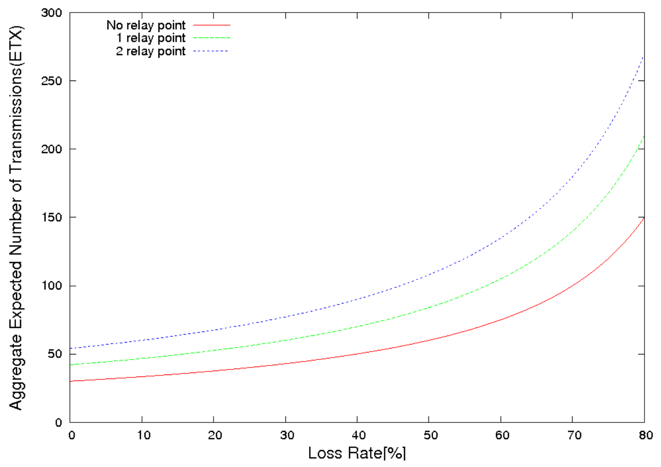


Figure 4.7: Total number of expected transmissions for a 3 × 4 grid as a function of packet loss probability p. Each mote delivers a single packet to the gateway. The different lines correspond to grids that require 0,1 or 2 relay points to connect to the network's gateway.

before it needs to recharge its battery. Hence, accurately estimating the robot's energy consumption is a prerequisite for the correct operation of the data muling system.

The robots we use are powered by standard, six-cell 7.6 V 3,000 mAh NiMH battery packs. The current drawn from the battery varies depending on the robot's activity. We classify these activities in three states: *stand-by, moving, and downloading/uploading data*. We experimentally measured the minimum and maximum current values for each of these three states. The measured values are: 560 $\sim$ 620 mA (stand-by), 630 $\sim$ 940 mA (moving), and 590 $\sim$ 750 mA (download/upload).

During our experiments, robots had an average speed of 0.2 m/sec. Downloading 11.4 KBytes of data from each mote took one minute in average, which consists of 20 seconds beaconing and actual data download time. In the first experiment, the trajectory length for one data mule was 46.2 m. The total time used by the robot to follow the trajectory, download data from the motes, and upload to the gateway was 16:51 minutes. In the second experiment, we used two robots where the robots' trajectory lengths were 35.5 m and 34.7 m. The total times were 9:56 min and 9:57 min, respectively. Finally, in the third experiment, we used three robots with trajectory

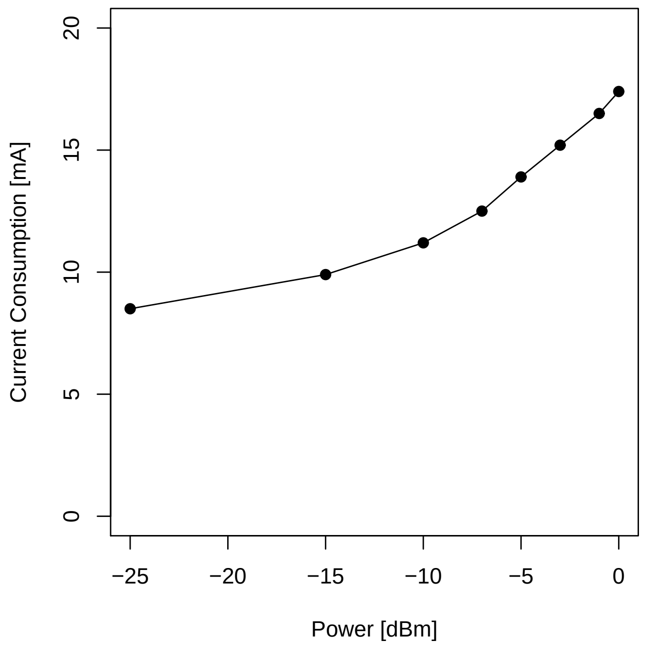


Figure 4.8: Current consumed by the mote's radio (TI Chipcon CC2420) as a function of transmission power.

lengths of 30.5 m, 30.6 m, and 29.3 m. The total times in that case were 8:33 min, 5:33 min and 8:26 min, respectively.

While calculating the energy consumption, we consider the maximum measured current values. According to this assumption, the robot consumed 1.69 Wh in the first experiment. In the second experiment, both robots consumed 1.02 Wh, while in the third experiment, robots consumed 0.87 Wh, 0.59 Wh and 0.86 Wh, respectively. One can then use these estimates and the battery's capacity (22.8 Wh), to determine how many tours the robots can complete before they need to recharge their batteries.

4.4 Efficient Strategies to Gather Data from WSNs

In the previous section, we showed the energy savings achieved by using robots to directly download data from sensors. Next, we address how we can further reduce the energy consumption due to beaconing and packet losses.

As described in Section 4.2.2, motes use beacons to check whether the robot is arrived or not. We could further reduce the mote's energy consumption if the mote knew the arrival time of the robot. In this case, the mote broadcasts a single beacon only shortly before the robot is expected to return. However, in a real system, a robots arrival to a sensors vicinity is not a deterministic process. Rather, it is a stochastic process where the arrival time is influenced from uncertainties in the navigation times. However, we can still reduce the number of beacons by adapting the beacon interval to match the robots arrival pattern. Consequently, we conserve energy and increase the mote's life-time. In our previous work, we address this problem and present an optimal beaconing schedule when the time of arrival is given as a probability distribution [119].

In this Section, we further improve the energy savings by minimizing the energy wasted due retransmissions. The quality of the communication link between a robot and a sensor can affect the the number of transmissions to download a fix amount of data. Due to various effects (e.g. multi-path effects) packet loss rate may vary inside the mote's communication range. We define a "good download location" as a location where the packet loss is minimal. If the robot can utilize its mobility and find a good location to download data, this can yield significant energy savings since retransmission is minimized. This statement is further justified in Section 4.4.1 where we address this search problem and present a data-driven strategy to find a good download location. In indoor environments where the behavior of the signal is unpredictable due to multi-path effects and the dynamic nature of the environment, it is easy to see that there is no online algorithm with provable performance guarantees[1] . Therefore, we present a heuristic strategy based on extensive experiments we performed to understand the effect of robot's location and orientation on the signal quality.

In Section 4.5.1, we present the details of a system implementation that utilizes

[1] For any given deterministic strategy, an adversary can pick the "good" location to be the last location visited by the strategy.

robots for gathering data, and demonstrate the utility of our algorithms with experiments run on this system.

4.4.1 Link Quality Experiments

When the robot is downloading data from a node, the quality of the wireless communication link is a crucial factor in determining energy spent in communication: when the link quality is high, the same amount of data can be transferred using less energy. This in turn, drastically affects the lifetime of the node. In the next section, we present a motion strategy for a robot to find a good location to download the data. The algorithm is based on insights from a series of experiments which we describe here.

We started our experiments by collecting data using the setup shown in Figure 4.9 where we placed an 11×11 grid on a $3m \times 3m$ indoor environment. We mounted a base station mote on our robot (iRobot Create with Asus Eee PC) and the robot autonomously visited grid points while pointing to a fixed direction.

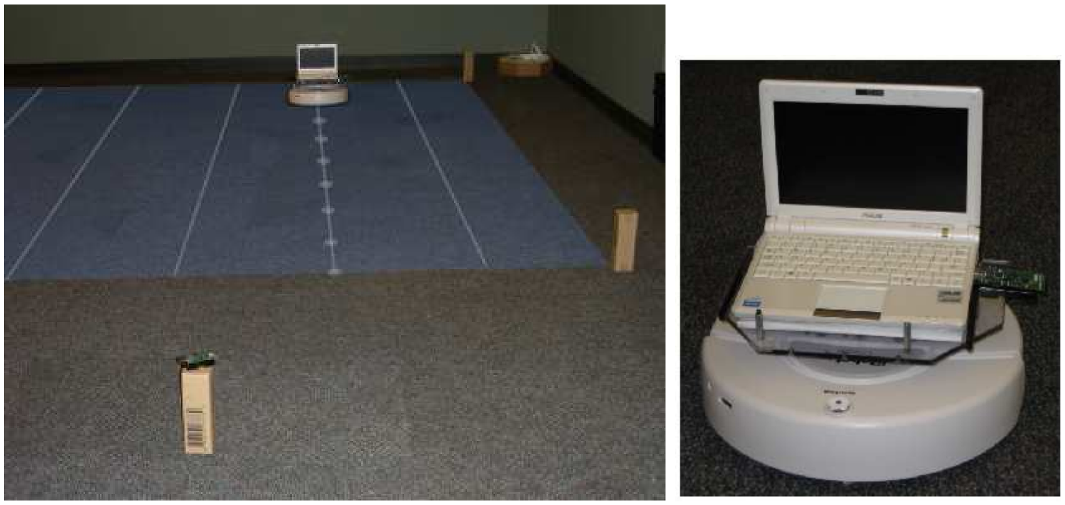


Figure 4.9: **Left:** Experimental setup to measure the link quality of data transfer from a mobile robot to a base station, with the robot moving on a uniform grid.

Figure 4.10 shows signal strength measurements on the grid when the mote was placed at (3,10). Each measurement was taken by sending a 4 byte message during which Radio Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI) values were recorded. From each location, robot samples 50 measurements. The top two plots in Figure 4.10 show the mean and median values of the LQI measurements. The bottom two plots show the RSSI values. All plots give a unified view of the link quality: although in general the LQI increases as we get closer to the mote, the surface is not smooth and contains many deep drops due to multi-path effects.

The next experiment illustrates the effects of the location and the link quality on the time to download the data. In Figure 4.11, the top figure shows the time to download 50 messages from each grid point. The peaks show a correlation with the deep fading effects in the previous experiment (Figure 4.10). Bottom left (resp. right) figure shows the relation between LQI (resp. RSSI) measurements and time transfer. As it can be seen in the left figure, the transfer time is very low for LQI measurements above 95. On the other hand, the transfer time increases drastically for values lower than 95. *This observation shows the potential utility of controlled mobility: robots can reduce the data download time (and increase the life of the sensor network) by finding a "good"*[2] *location to download the data.*

The next experiment sheds further light on path-loss and multi-path affects on link quality. To cover a wider range, we moved the robot on a line-segment in a corridor in our building and placed a mote on the mid-point of this line segment. In Figure 4.12, the mote is located at $x = 26$ and robot starts taking measurements at $x = 1$ and ends at $x = 51$. The discretization level is 1 foot, the robot takes 50 measurements from each location. Top figure shows mean values of 50 measurements, bottom figure shows median of the measurements. As expected, the link quality increases when robots get closer to the mote, while it tends to decrease while getting further away from it. After performing similar experiments, we concluded that the following observation explains the link-quality behavior better: *Within a certain range (±8 ft, in this case), the link quality is consistently "good" and unpredictable (random) outside this range.*

Most robotic systems have a rotational component which means that we can control the orientation of the robot. Therefore, the robot should search for not only a good location but a good orientation as well. The next experiment focuses on this aspect. Figure 4.13 shows the change in link quality with the various orientations of the base station (on the robot) at fixed locations. The figure shows the results for 4 fixed points

[2] *In this case, a good location is one where the LQI value is greater than* 95.

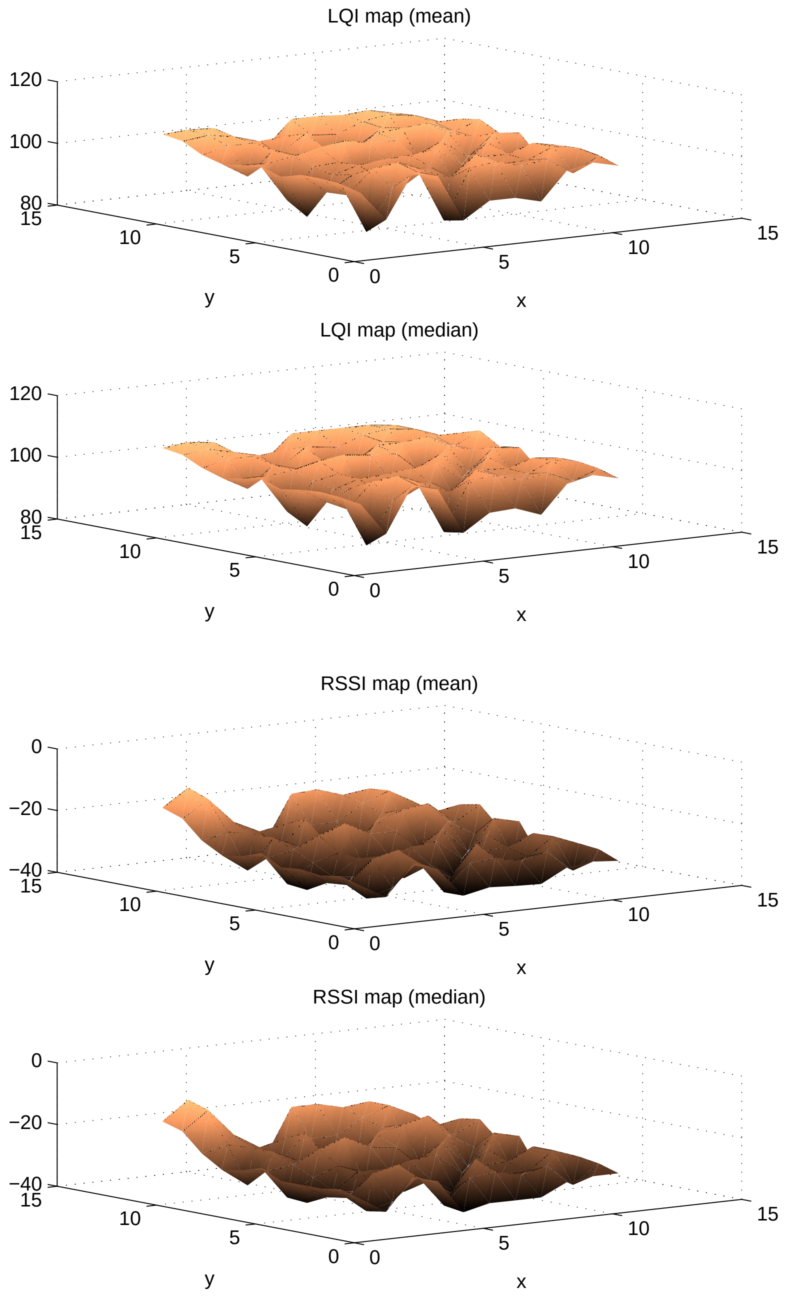


Figure 4.10: The robot visited each location, and took 50 measurements. From top to bottom: mean and median values of the LQI measurement, mean and median values of the RSSI measurements.

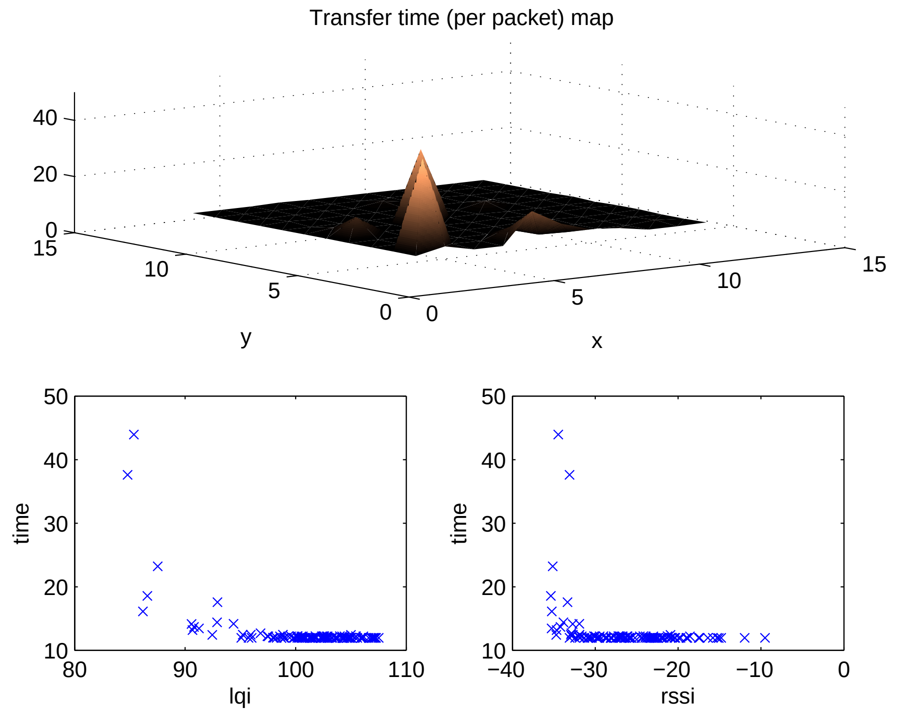


Figure 4.11: Time to download 50 messages from each grid point as function of location (top), and as functions of LQI (bottom-left) and RSSI (bottom-right). With controlled mobility, the robot can decrease the download time significantly by moving slightly.

at distances 5,10,15 and 20 feet from the mote. The results show that when the base station is close to the mote, the orientation does not affect the link quality significantly. However, when the distance is large, small changes in orientation may result in drastic changes in link quality. Moreover, this change is not easily predictable. For example, when the robot is at the furthest point (the 20-feet curve), mote and base station point towards each other when the angle is 180°. In this orientation, the LQI is 95. If robot turns 45° in counter-clockwise direction, the LQI value increases to 100. If the robot turns 45° more on counter-clockwise direction, then the LQI value suddenly drops to 80. This example also shows that measurements from various orientations may not give a clear indication about the direction of the mote.

The results of these experiments can be summarized as follows:

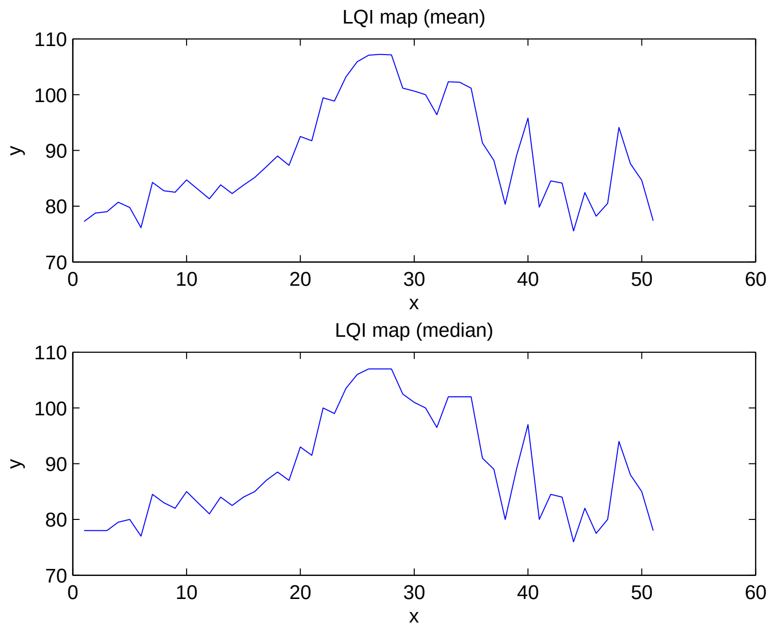


Figure 4.12: The mote is located at $x = 26$. The robot moves from $x = 1$ to $x = 51$. Within a range of ± 8 ft, the link quality is consistently "good". It is unpredictable (random) outside this range.

- Within a certain distance (an environment dependent parameter), the signal quality is predictably good and the orientation of the robot does not make a significant difference.

- When the robot is outside this range, it is very difficult to use local information (such as gradient) to find the location or orientation of the mote.

In the next section, we present a search strategy based on these observations.

4.4.2 Local Search Algorithm

In this section, we describe a search algorithm to find a good download location. In many applications, it is beneficial to search for this location in an online fashion because the location of the mote whose data will be downloaded can change locally, the signal

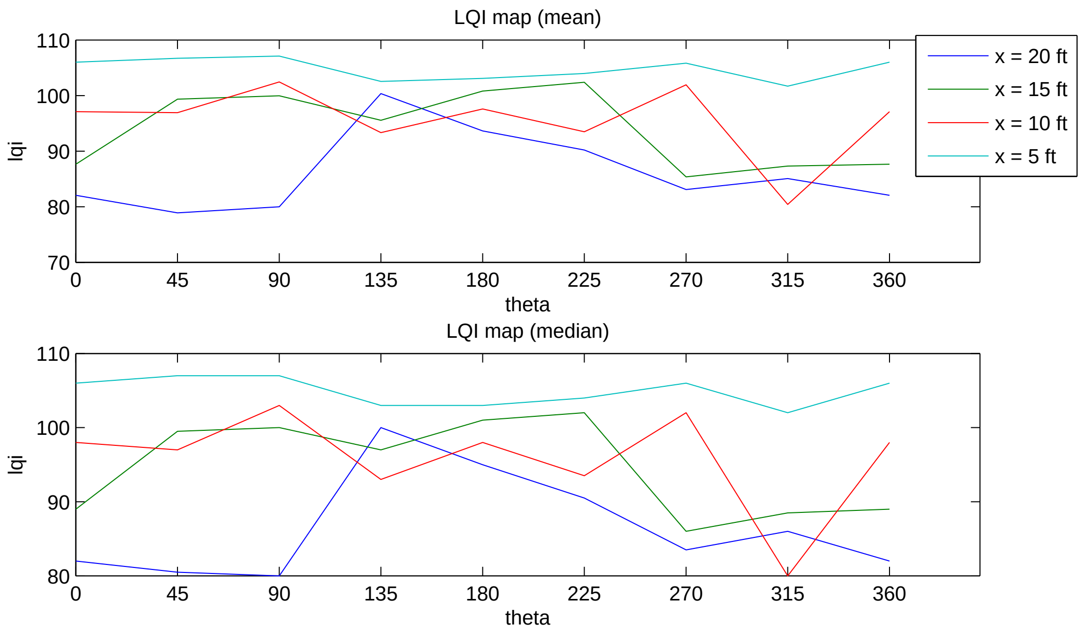


Figure 4.13: The θ-values correspond to robot orientations. Each curve corresponds to a different location on a line. The mote is located at $x = 0$. The behavior of RSSI or LQI as a function of rotation is not easily predictable.

properties may change over-time, or the robot may not have the localization capability to visit a location accurately.

The experiment in the previous section indicates that it is very difficult, if not impossible, to use local gradient information to seek a good location. A global approach is needed. The strategy we present uses two environment dependent parameters. The first parameter β is a lower-bound for an acceptable signal strength (LQI value). For example, an appropriate β value for the environment where the experiment shown in Figure 4.11 was performed, is 95. The second value α is mainly a grid resolution and it is set to the distance within which the link quality is predictably good. For the environment where the experiment shown in Figure 4.12 was performed, an appropriate α value is 8 ft.

Upon hearing a beacon message, the robot searches for a good location by placing a grid on the environment where the dimension of each cell is determined by α. When the robot visits a grid cell, it rotates $0, 90, 180, 270$ degrees. This allows us to get rid of local multi-path effects and to simultaneously seek a good orientation. At each rotation, the robot takes five link quality measurements. The quality of each orientation is defined as the median of these five measurements. The weight of each cell is then set to the the highest of these four median values. In the algorithm below, $measure(c)$ subroutine performs these steps at cell location c. We also keep track of a table where we store the expected link quality values. For each unvisited cell, we set the *expected_weight* value to the average link quality value of its immediate neighbors. Next, robot visits the location with maximum expected weight. The algorithm is given below:

Algorithm 2 LocalSearch

Cell Discretization: C

```
1: $expected\_weight(c \in C) \leftarrow 0$, $c \leftarrow (0,0)$ (initial location)
2: while there are unexplored cells do
3:   if $measure(c) \geq \beta$ then
4:     return
5:   end if
6:   Forall $c' \in Neighbor(c)$ if $c'$ is unvisited,
     $expected\_weight(c') = mean(\forall_{c'' \in Neighbor(c')} measure(c''))$
7:   $c \leftarrow \max_{c'} expected\_weight(c')$
8: end while
```

Remark 18. *If the β value is not known, we can set it to a high value. In this case, the robot will visit all grid-cells. We can then pick the best location.*

Remark 19. *We can incorporate collision avoidance into the strategy by setting the weight of a cell to zero if there is an obstacle at that cell.*

In the next section, we demonstrate the utility of this strategy with a series of experiments.

Local Search Experiments

We tested the search algorithm in a number of settings. In this section, we present two of these results.

The first experiment was performed in the indoor setting shown in Figure 4.14-left. The signal strength level in TelosB mote was set to 3. The other system parameters were $\alpha = 1.5m$ and $\beta = 100$.

When the robot started from the initial location shown in the figure, it quickly converged to a good location. Robot's steps are shown in Figure 4.14-left bottom where the visited cells are labeled with the format (s, r): s is the order the cell was visited and r is the maximum median value sampled from four orientations.

The second experiment was performed outdoors with $\alpha = 3m$ and $\beta = 100$. As shown in Figure 4.14-right, the robot quickly converged to a good location. Link to the videos of all experiments can be found in Appendix A-Multi Media Extension No: 3.

In conclusion, the simple search strategy presented in this section was efficient in finding a good download location. Where most local search heuristics would get stuck with a single cell, the presented search strategy quickly converges to a robot pose from where the data can be downloaded efficiently.

4.4.3 System Design

We performed experiments to demonstrate the utility of incorporating both beacon scheduling, and local search. Here, we describe the design of the data mule system used in the experiments.

Our system consists of three classes of devices. (i) The sensor motes are Crossbow Telos, (rev. B) [2] which use the CC2420 chip. They are IEEE 802.15.4 compliant. We deployed three static motes in the fourth floor lounge of the Digital Technology Center (DTC) at the University of Minnesota, Twin-Cities. The experimental setup is shown in Figure 4.15. (ii) The mobile robot is an iRobot Create without the command module. (iii) The control program for the robot runs on an Asus Eee PC, which interfaces with the Create directly through a USB-to-Serial cable. The system ran Linux (Ubuntu) and our Java and C++ programs used serial communication libraries to write motion commands to the robot, in accordance with the Create Open Interface (OI) specifications.

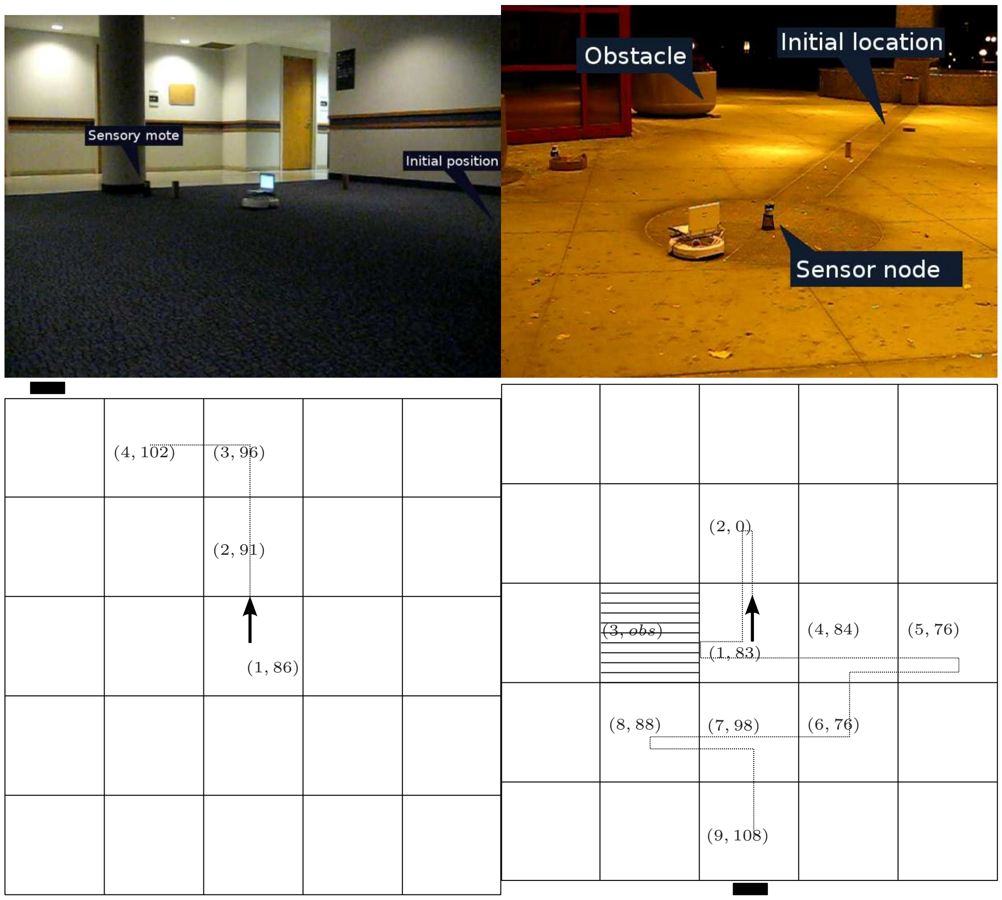


Figure 4.14: Bottom figures show a virtual grid used by the search algorithm. The visited cells are labeled with the format (s, r): s is the order the cell was visited and r is the maximum median value sampled from four orientations. The black rectangles show the location of the mote. **Top Left:** The setup for an indoor experiment. The picture shows the best configuration found by the search algorithm. **Bottom Left:** Steps in finding a good location in the setup shown on top. **Top Right:** Search performed in an outdoor setting. The picture shows the best configuration found by the search algorithm. The shaded cell corresponds to the obstacle that robot avoided. **Bottom Right:** Steps taken during outdoor search.

4.4.4 Experiments

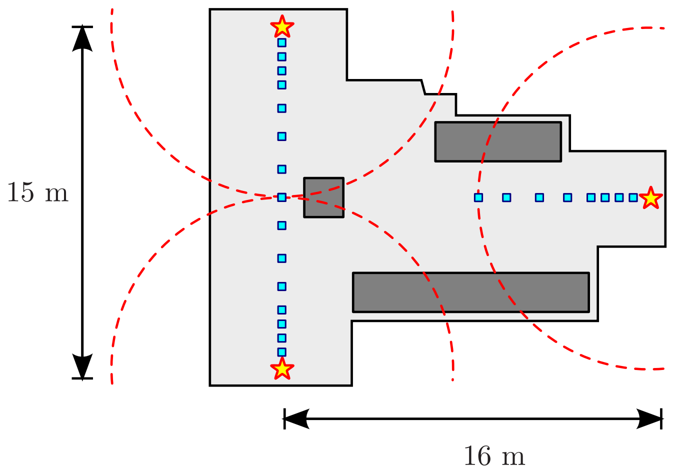


Figure 4.15: A proof-of-concept deployment. The stars are approximate locations of the data nodes. The dashed lines show their communication range. The squares are locations where the robot starts either the download or the local search.

We performed four experiments to demonstrate the utility of incorporating both beacon scheduling (presented in [119]), and local search. The experiments are: baseline (B), beacon scheduling (BS), local search (LS) and both local search and beacon scheduling (LSBS). Each experiment consisted of 8 rounds. In each round, robot visits a pre-defined location for each mote, and downloads the data from that mote. The locations that the robot starts downloading (shown as squares in Fig. 4.15) are fixed for comparison purposes: For example, if in experiment B the robot downloads from a fixed location then in experiment LS, the robot starts the local search from the same location.

We picked a range of download locations in a mote's vicinity to simulate the effects of localization uncertainty: If the robot does not have accurate means of localization, even if it targets a fixed location to download data, it may be off from that location by a distance given by the uncertainty range. After arriving at a predetermined location, the robot either directly downloads the data (experiment B and BS), or performs a local search to find a good location before downloading (LS and LSBS experiments). After download finishes, the robot either continues to the next mote directly (experiments B

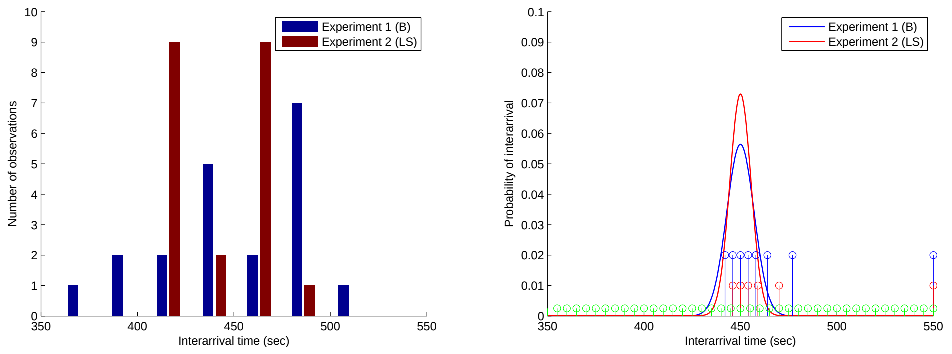


Figure 4.16: The robot interarrival times from our experiments were modeled as normal distributions.

and LS), or computes an updated beacon schedule based on interarrival times, uploads it to that mote and proceeds to the next mote (experiments BS and LSBS).

In all experiments, the beacons are special messages whose payload consists of (i) the node id of the mote, and (ii) a sequence number of the triggered beacon. To compare the local search with base case, we needed a mechanism to compare the trade-off between energy gain in efficient download and energy spent in extra beacons sent during the search phase. Therefore, we used data packages which are the same as the beacon type messages. To download the data on the mote, the robot must successfully hear 100 additional beacons. This represents scenarios where the data stored on the mote corresponds to 100 messages and all of it must be successfully downloaded. This way, we can use the last received beacon sequence number for each mote to represent the total energy consumption metric spent in beaconing and download. Even with this modest amount of data, each experiment lasted about an hour.

In experiment B, we chose the beacon interval for discovery phase as 5 seconds. This guaranteed an expected robot waiting time of 2.5 sec. The optimal beacon scheduling algorithm of [119] and the robot inter-arrival distribution observed in experiment B are used to achieve 2.5 sec waiting time in experiment BS. In experiment LSBS, we used the interarrival times from the LS experiment to compute the optimal beacon schedule for this case. The recorded interarrival times and the robot's arrival model are shown in Figure 4.16. In experiment BS, the optimum beacon schedule uses 8 beacons.

In comparison to the number of beacons (550/5 = 110) in the base experiment B, the beaconing strategy yields significant energy savings (8 beacons) while satisfying the same expected waiting time constraint. Comparing the total number of beacons, we can see the effect in total performance. Beacon scheduling in experiment BS reduced the total number of beacons to 2791 compared to 4839 in experiment B, the baseline (first and second columns in right of Table 4.1).

The left one of the two tables shown in Table 4.1, shows the packet loss rates for various locations in each experiment. Clearly, local search provides a significant reduction in packet loss rate for the first two locations (compare B versus LS and BS versus LSBS) where the distance prevents a lossless communication between the mote and robot. For example, in the experiment B, the first mote has to sent 538 packets until the robot successfully downloads all of the 100 data packets, whereas after local search no packet is lost. We can see the efficiency of local search for the first two rounds of the examples in Table 4.1-right. On the other hand, for the rest of the rounds, the local search does not provide significant gains. In fact, the energy consumption slightly increases in experiment LSBS compared to BS experiment due to the overhead (i.e. number of beacons sent during local search).

It is worth noting that in this indoor setting, the robot's total path is relatively short compared to the search distance. Thus, the search overhead becomes comparable to the number of discovery beacons. When the travel distances are large, (e.g. in outdoor settings), the search overhead will become negligible. In this case, local search will yield more significant energy savings.

Overall, the experiments clearly demonstrate that (i) adaptive beaconing strategies yield significant savings in the number of discovery beacons sent, and (ii) local search strategies can result in drastic improvements in the download time when the link quality is unpredictable.

4.5 Opportunistic Path-planning under Disk Model

In our recent work [25], we introduced a path-planning problem called *Data Gathering Problem (DGP)*. In DGP we are given n sensor locations, k robots and a communication radius r. We assume that sensor's communication range is modeled as a disk which is

Dist.	B	BS	LS	LSBS
7.5m	173%	36%	0%	1%
6.25m	7%	21%	0%	1%
5m	11%	0%	0%	0%
3.75m	8%	0%	0%	0%
2.5m	2%	3%	0%	3%
1.7m	0%	0%	0%	0%
0.9m	0%	0%	0%	0%
0.1m	0%	0%	0%	0%

	B	BS	LS	LSBS
1-2	1642	897	997	828
3-8	3197	1894	3215	2152
Total	4839	2791	4212	2980

Table 4.1: The **Left** table shows the package loss rates for each experiments (B:Base,LS:Local Search,BS: Beacon Schedule, LSBS: Local Search and Beacon Schedule together) with respect to the distance that robot starts to download or starts to the local search. For each download we calculate the number of packet loss until robot hears 100 beacons. **Right** figure shows the total number of beacons send from 3 motes during the experiment.

centered at the sensor location and has radius r. The cost incurred by a robot tour is defined as the sum of the time robot spends while traveling and the time it spends to download data from the sensors. The goal in DGP is to find k tours which gather data from all sensors such that the maximum of the tour costs is minimized.

In [25], we presented an approximation algorithm which uses techniques developed for TSP with Neighborhoods (TSPN) and TSP with k salesperson (k-TSP) to obtain an approximation ratio of $e + 2 - 1/k$ where e is the approximation ratio for TSPN.

In this section, we present a modification of the DGP algorithm which alleviates some of the challenges faced when executing it on a real system. In particular, there are two major challenges: First, the sensing and actuation errors make it difficult to precisely follow the line segments output by the DGP algorithm. When the robot steers off the computed path, it can be within the communication range of a sensor other than the next sensor it is headed toward. Second, the disk model used for communication is not always accurate. One can choose the disk radius conservatively so that the expected signal quality inside the disk is good. However, occasionally it is possible to receive a good signal outside the disk. In Section 4.5.2, we make this observations explicit by giving real examples from field experiments.

We now present a modification of the DGP algorithm which opportunistically downloads from all sensors within the robots communication range (Algorithm 3).

Algorithm 3 *OPPORTUNISTIC_DGP_ALGORITHM*

```
1: Let $S$ be the set of sensors assigned to robot $u$ by DGP algorithm
2: while $S \neq \emptyset$ do
3:   Let $s_i$ be the next sensor to visit
4:   if $u$ hears a good signal from $s_i$ or close enough to $s_i$ then
5:     Stop and collect data from $s_i$ and set $S \leftarrow S/s_i$
6:   else if $u$ hears a good signal from a sensor $s_j$ such that $j >= i$ then
7:     Stop and collect data from $s_j$.
8:     Set $S \leftarrow S/s_j$ and remove the visit of $s_j$ from the DGP tour
9:   else
10:    Continue the tour returned by DGP algorithm
11:  end if
12: end while
13: Visit $s_1$ for the next tour then Goto 1
```

Let $S = \{s_1, \dots s_n\}$ be the ordered list of sensors assigned to a robot u after the execution of the DGP algorithm. We assume s_1 be the base station so that robot starts its tour from the base in each tour. Let s_i be the next sensor to visit in S. When running the opportunistic algorithm, the robot polls the sensors after s_i and checks if they are within the communication range. Most WSN systems include low-level support for scanning the communication channel. In our implementation, we utilize this capability of TelosB motes. If the robot hears from any sensor s_j such that $j \geq i$, it stops and collects data from s_j. The robot removes s_j from S, and continues toward s_i. After visiting the last node in S, the robot returns to s_1 ignoring all sensors heard in the mean time because their data has been downloaded recently.

In order to demonstrate the feasibility and utility of using robots for data collection, we developed an autonomous data muling system. In the next section, we present the system components and details of our implementation. In Section 4.5.2, we present

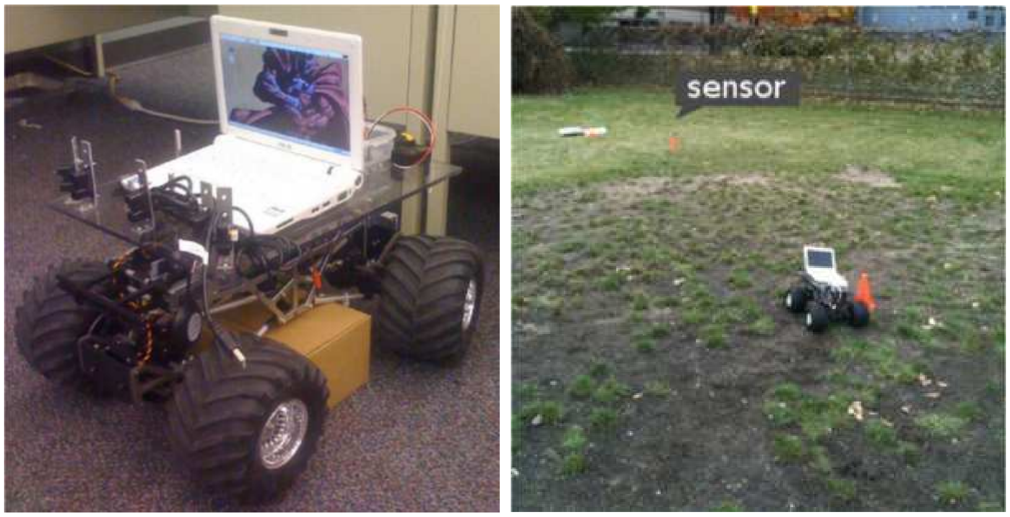


Figure 4.17: **Left:** Cyclops robotic platform developed for data muling. **Right:** A data mule system together with sensors.

results from field experiments.

4.5.1 System Design

In this section we describe the hardware and software components of our system.

Hardware

Since sensor networks are typically composed of inexpensive components, we focused on developing an inexpensive yet robust and effective system so that it can be used commonly e.g. in environmental monitoring. Considering these design choices, we developed an outdoor robotic platform which we call "cyclops" (Figure 4.17).

The entire system is composed of the following equipment and devices:

- Robotic data mule:
  - Radio controlled racing truck base (Tamiya TXT-1)
  - Motor driver (Novak Super Duty XR)
  - Micro controller (Robostix)

  - Laptop computer (Asus Eee PC)
  - Infra Red (IR) sensors (Sharp GP2Y0A02YK)
  - Digital compass (Sparkfun HMC6352)
  - GPS (BU-353)

- Environmental sensors (Crossbow TelosB motes)

The cyclops is based on an RC truck with front servo steering. In order to increase its maneuvering capabilities, we installed a back servo motor which steers in the back wheels in the opposite direction of the frontal wheels. A robostix micro controller is interfaced with the motor driver and steer servo motors to control the robot. For obstacle avoidance we placed four IR sensors in front. One sensor is pointed directly forward whereas two sensors look 45 degrees to the sides. This configuration of IR sensors allows the robot to cover a large range of frontal view. We also placed a fourth sensor looking down to act as a cliff sensor.

We use a compass and a GPS for estimating the robot's pose during navigation (We present the details about localization and navigation in the next section.). Compass and IR sensors are interfaced with a robostix controller. We use a laptop for computing and executing the motion plan. The laptop is interfaced with robostix through serial port. It sends drive commands and receives sensor reading through this serial port. In addition, a GPS device is directly attached to the laptop's serial port. There are three power sources on the robot. The first one is a 12V battery which powers the motors. Robostix controller and the laptop run on dedicated batteries. The battery life of the entire system is approximately two hours indoors. However we observed that this time could drop to half an hour outdoors depending on the environment. Even though this is sufficient for field tests, a longer battery life is desirable for real-life deployments. We are currently investigating the use of solar power to address this issue.

Finally we use TelosB motes as stationary sensor devices. TelosB motes are equipped with a USB interface, an IEEE 802.15.4 radio, a low-power CPU and on-board sensors (temperature, light and humidity) which are powered by two AA batteries. Using low-power sensing, these wireless sensor devices make it feasible to monitor environments for months. We use motes for two purposes. Motes of the first type are called *sensor motes* which are deployed onto the field and collect data from the environment. The

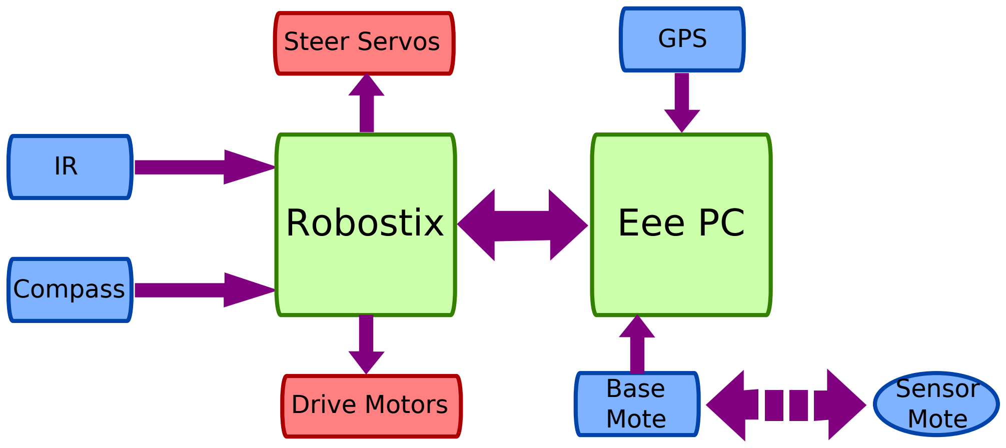


Figure 4.18: System diagram. Robostix controls drive and servo motors, reads data from IR sensors and compass. Eee PC communicates through robostix to control the robot and reads data from GPS. Base mote downloads data from sensor motes and sends the data to the laptop.

motes of the second type are called *base motes* which are attached to the laptops on the robots and used for enabling the communication between sensors and the laptop. A base mote communicates with sensor motes using ZigBee protocol and forwards the messages to the serial port of the laptop and vice versa to enable this communication. A diagram of our overall system is shown in Figure 4.18.

As discussed earlier, we made our design choices so as to construct an inexpensive and robust robotic platform for environmental monitoring applications. Cyclops has light aluminum ladder frame and multi-link suspension to make it possible to navigate in rough domains with high speed (max. 6 kmph) while carrying all the payload (about 7 kg including the base platform). Moreover including all the equipment and devices, the cost of the entire system (not including sensor motes) is about $1,000.

Navigation Algorithms and Software

The software portion of our system consists of two components. The first component is an embedded program developed for robostix. This component is implemented in C++ and provides an Application Programming Interface (API) for controlling the robot and reading from compass and IR sensors. The second component is developed for the laptop and used for high level control (navigation and sensory data download). Since we have various sensors which have to be read in parallel, we implemented a multi-threaded system. A separate thread reads and writes from/to each device (compass, GPS, robostix and base mote) and updates its state in the main thread. Since Java provides an advanced threading scheme, we used Java to implement the high level application. We also implemented a server/client application and a GUI to visualize the robot's state from a remote computer.

Motes are programmed in nesC. Sensing motes send simple beacon messages, while the base mote receives these messages and filters them according to their mote ids. Messages coming from the target mote are processed in the base mote. The base computes the link quality indicator (LQI) values of the packets received and sends them as serial packets to the Java program running on the laptop. The java program saves the timestamps (start and end time of the download) and packets received.

The DGP algorithm in [25] yields way-points to be visited along with the ids of sensors whose data must be downloaded at each way-point. We now describe the algorithm for navigating between these way-points.

The robot uses only compass and GPS for pose estimation. Since neither sensor provides accurate information, the robot has to deal with large uncertainties during navigation. At the beginning of its tour, the robot executes an initialization routine to ensure that it receives the most accurate readings from these devices.

The compass requires an initial calibration step to adjust its values according to the earth's magnetic field and external electro-magnetic fields. For that purpose, robot loops around a circle twice at the beginning and calibrates its compass. Afterward the robot uses only compass measurements to determine its heading.

It is well-known that GPS performance changes according to external factors (e.g. environmental effects, and the number and configuration of satellites). To improve its accuracy, we enabled Wide Area Augmentation System (WAAS) feature of our GPS device which corrects for GPS signal errors caused by ionospheric disturbances, timing, and satellite orbit errors. Moreover, we determined that GPS accuracy is higher when robot is moving compared to when it is stationary. Hence, initially to get a better estimate, the robot moves on a straight line until the variance of the heading measurements drops below a threshold. After good GPS measurements are obtained, the robot computes its desired heading by using its current location and the target location (next way-point). The robot uses its compass to move in the desired direction. However, due to errors, sometimes it goes off of the desired trajectory. The opportunistic algorithm presented in this section overcomes some of the inefficiencies associated with this type of error. When the error goes beyond a threshold, the robot re-computes its trajectory to the next node. In experiments described next, the uncertainty of GPS measurements was about 3 meters.

4.5.2 Experiments

In this section we present field experiments conducted by the system discussed above. These experiments are designed for two reasons. First we aim to show the practical feasibility of using an inexpensive robot as a data mule for collecting data from sensors. Second we aim to show how our algorithm can be improved in practice under navigation and communication uncertainties. For this purpose we conducted two sets of experiments. In the first experiment we showed the feasibility of our system using DGP algorithm presented in our recent work [25]. Using observations from the first experiment, we extended our algorithm by using the opportunistic approach presented at the beginning of the section. Since we have only one robot, our experiments do not involve multiple robots. However, since the DGP algorithm partitions the sensors among the robots and robots do not communicate during the data collection, in a single robot is sufficient to accomplish our goals.

All experiments were conducted in East River Flats near the Mississippi River/Minnesota. The ground surface of this field is mainly covered with long grass which makes it a challenging environment for navigation. See Figure 4.19. We placed seven sensors ($S = \{S0, \dots, S6\}$) in a rectangular area of approximate size of 170m$\times$70m. Since a base and a sensor are treated as identical disks, we did not deploy a specific base. Instead, we treat the last sensor as the base station. Sensors' communication range was set to 6m. In our deployment, the communication disks of sensors S0, S1 and S2 overlap as well as the communication disks of sensors S4 and S5 which were placed at the top of a hill.

We computed a tour for this deployment using the *SparseTspnTour* algorithm presented in [25]. *SparseTspnTour* chose $S0$, $S3$, $S4$ and $S6$ as sensors with non-overlapping

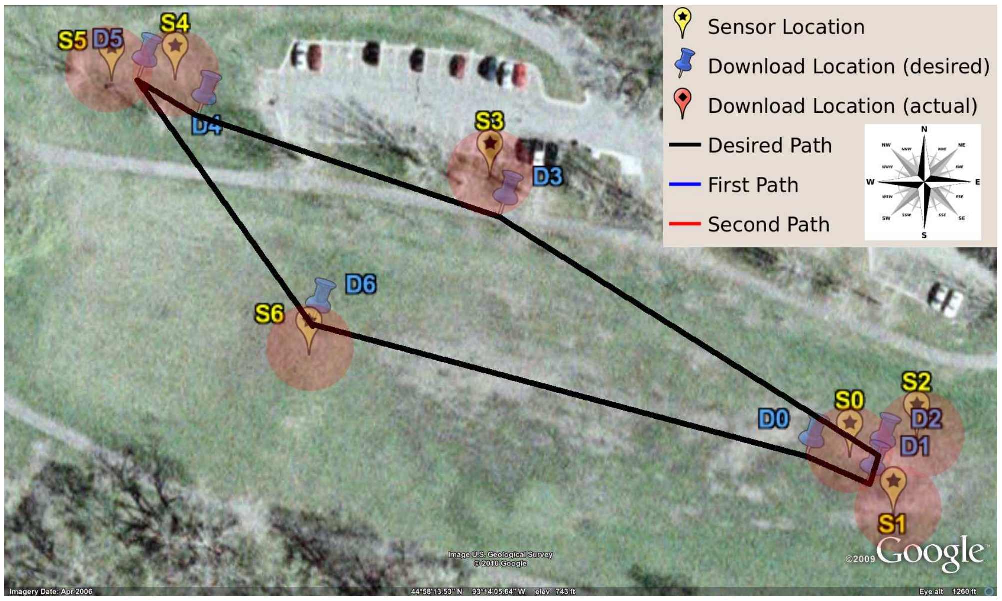


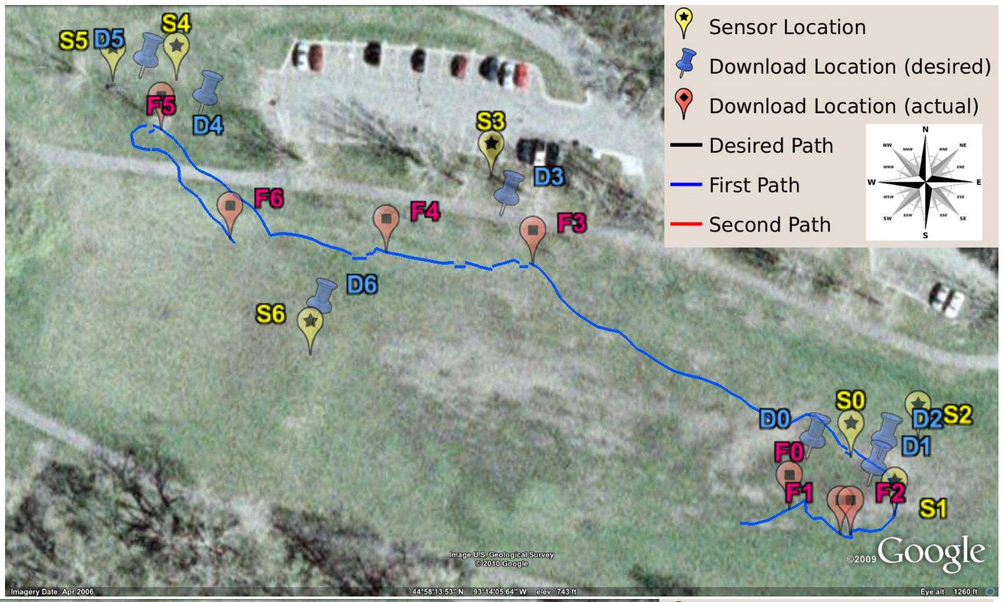
Figure 4.19: Field deployment in East River Flats: $\{S0, \ldots, S6\}$ are sensor locations. $\{D0, \ldots, D6\}$ are the respective download locations computed by our DGP algorithm. Red (shaded) disks show the communication disks and black tour shows the ideal TSPN tour.

communication disks in I. For each overlapping sensor we compute a download location. The ordered tour is given by: $\tau = \{D0, D1, D2, D3, D4, D5, D6\}$. The robot was programmed to download twenty packets of size 32 bytes from each sensor where each sensor was programmed to send packets as beacons with time interval 250 milliseconds.

In the first experiment we made the robot execute the tour τ. Due to GPS uncertainties we programmed the robot to stop if its distance to destination point Di is closer than some threshold. Moreover during our initial experiments we realized that the robot occasionally hears good signal from long distances while moving towards the target sensor due the environmental effects. For example robot was able hear the sensor at the top of the hill ($S4$) from approximately 25m distance. In these cases robot does not have to navigate all the way to the desired download location. Hence we modified our algorithm by adding the following condition: robot collects data from sensor Si if it is close enough to Di or it hears a good signal from Si. Note that this modification does not allow downloading from nodes out of order. It simply changes the download location if a better signal becomes available early on.

Figure 4.20 show the GPS trace of the robot during the first (top image) and the second (bottom image) tours respectively. The actual download locations were marked with the different placemarks and labeled as $\{F0, \ldots, F6\}$. The actual and intended download locations were different due to the uncertainty in GPS readings and the modification regarding the download location. The download, travel and total times are reported in Table 4.2. We also include the total tour time as the sum of download and travel times.

Since the actual starting positions in this experiment and the next one were different, we started the first tour from $S0$. Hence the travel time to reach $S0$ was not counted in the first tour. However in the second tour the travel time from $S6$ to $S0$ was counted as the travel time to reach this sensor. Observe that the download times are consistent with each other except the last sensor in the second tour. This is because, while moving towards $S6$ robot heard a temporary good signal from $S6$ and stopped. However the signal strength dropped during the download which extended the download time.



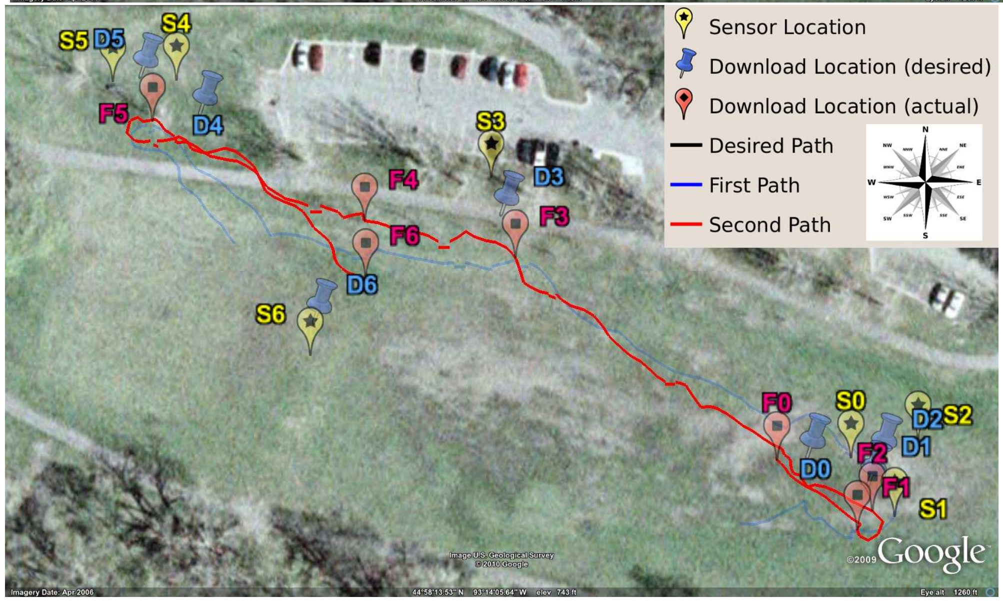


Figure 4.20: **Top:** First tour by the robot (blue trace). Fi is the actual location from where the robot downloads the data from Si for $i = 0, \ldots, 6$. **Bottom:** Two complete tours of the robot where first tour is marked in pale blue and the second tour in red. These images are best viewed in color.

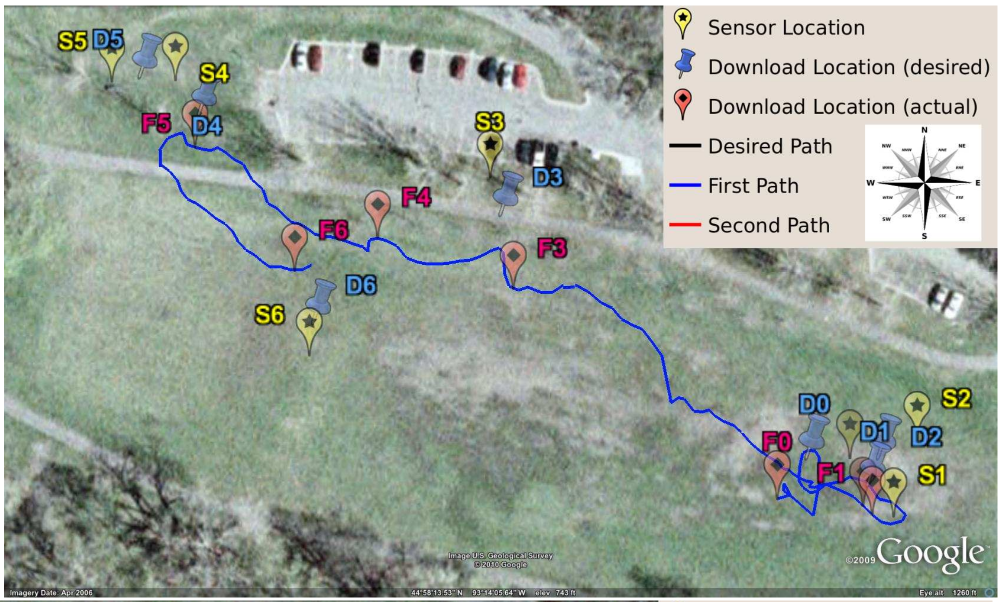
In the first experiment, we realized that the robot hears from other sensors while

	Tour-1		Tour-2	
	Download time	Travel Time	Download Time	TravelTime
S0	4.9		5.3	151.1
S1	5.1	12.2	5.9	26.0
S2	5.1	1.2	4.8	0.9
S3	5.0	100.0	5.1	113.7
S4	4.9	26.7	4.9	33.9
S5	4.8	64.5	4.8	57.3
S6	5.0	41.7	30.0	100.5
Total	34.84	246.2	60.93	483.33
Tour Total	825.3			

Table 4.2: Download and travel times in seconds from the DGP experiment. The first row of Tour-2 includes the time to travel from S6 to S0.

moving towards target location. This specially happens when robot is collecting data from overlapping sensors. For example, when robot is moving towards $S1$, sometimes it hears a good signal from $S2$ before $S1$. In this case robot can download data from $S2$ and therefore can eliminate the traveling cost to $S2$. Hence we proposed an opportunistic version of the DGP algorithm which is presented in detail in Algorithm 3. In contrast to download location modification, this modification allows robot to download from any node that the robot can hear which consequently changes the order of the tour.

Figure 4.21 show the GPS trace from the opportunistic DGP experiment. The download and travel times during this experiment are reported in Table 4.3. In this experiment robot leverages the opportunistic approach to reduce its tour time. The total travel time in the first experiment was 730 seconds whereas in the opportunistic DGP experiment the total travel time was 708 seconds. However it is hard make a reliable comparison between the two strategies since the actual paths and the actual download locations were different. On the other hand we can see the advantage of the opportunistic approach when robot downloads data from $S2$ in the first tour. Since $S0, S1$ and $S2$ were close robot heard a good signal from $S2$ when moving from $S0$ to $S1$ and download the data on the way. Hence it did not spend extra time to visit $S2$ which otherwise it would have spent about 13 seconds to navigate to $D2$ as in the second tour.



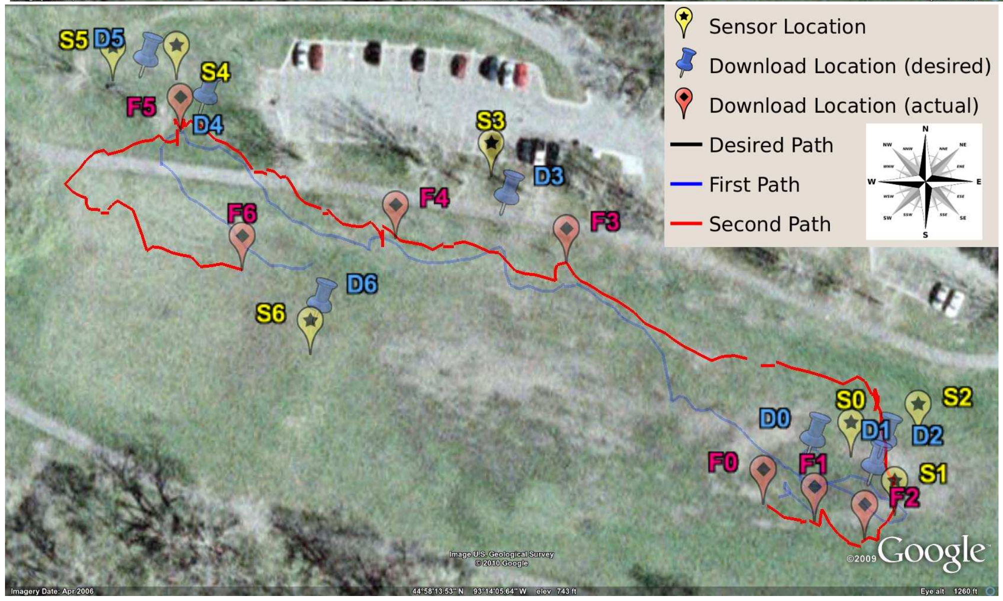


Figure 4.21: **Top:** First tour from the opportunistic DGP algorithm (blue trace). Fi is the actual location from where the robot downloads the data from Si for $i = 0, \ldots, 6$. **Bottom:** Two complete tours of the robot where first tour is marked in pale blue and the second tour in red. These images are best viewed in color.

	Tour-1		Tour-2	
	Download time	Travel Time	Download Time	TravelTime
S0	4.8		5.0	97.5
S1	4.8	46.6	4.7	8.1
S2	5.0	0	5.3	13.4
S3	4.8	113.3	6.3	111.4
S4	5.8	28.5	5.7	35.0
S5	4.9	47.6	8.2	72.5
S6	5.9	42.8	5.6	91.2
Total	36.08	278.75	40.93	429.07
Tour Total	784.82			

Table 4.3: Download and travel times in seconds from the opportunistic DGP experiment. The first row of Tour-2 includes the time to travel from S6 to S0.

Next we compute how much the robot deviates from the ideal solution shown in Figure 4.19. The length of the ideal tour is 550m. The length of the robot's path in the first experiment was 625m. In the second experiment (opportunistic) this length was 629m. Therefore the uncertainties increase the tour length by 13.6% and 14.3% respectively. In terms of download times ideal solution would have spent at least 65.8 seconds to download data from all the sensors (assuming 4.7 seconds download time from each sensor). On the other hand in the first and the second experiments the total download times were 95.8 and 77 seconds, respectively. These yield to 45.6% and 17% increase in the download times. In terms of total tour times ideal solution would spend 615.8 seconds to finish two tours assuming a 1 m/sec average speed. The total tour times in the first and the second experiments were 825.3 and 784.8 seconds respectively. Hence the overall deviations from the ideal solution were 34% and 27.4%. Link to the videos of all experiments can be found in Appendix A-Multi Media Extension No: 4.

Although the number of experiments conducted were limited, the results show that despite the navigation and communication uncertainties, the performance of the system was not too far from the ideal solution. More importantly, they demonstrate that a data muling system built from inexpensive off-the-shelf components is feasible. In the future we aim to reduce navigational uncertainties by improving our algorithms and hardware capabilities.

4.6 Path-planning under Two-Ring Model

The uniform disk model ignores the fact that download time is a function of the signal strength. Recent experimental work [26], showed that the following *two ring communication model* models the dependency of communication quality on signal strength accurately in most sensor network deployments. In this model, there are two concentric disks centered at the sensor location. Inside the inner disk the communication is reliable thus the expected download time is shorter. Between the boundaries of the inner and the outer disks, communication is possible however due to increase in packet loss rate, the expected download time increases (see Figure 4.23). Collecting data under *two ring communication model* is a new optimization problem which we call *Two-Ring Tour Problem* (TRT). In this version of the problem the tours are no longer optimal TSPN paths. An optimal tour is the one which trades-off between downloading from the outer disk or going further to download from the inner disk with shorter download time (see Figure 4.22).

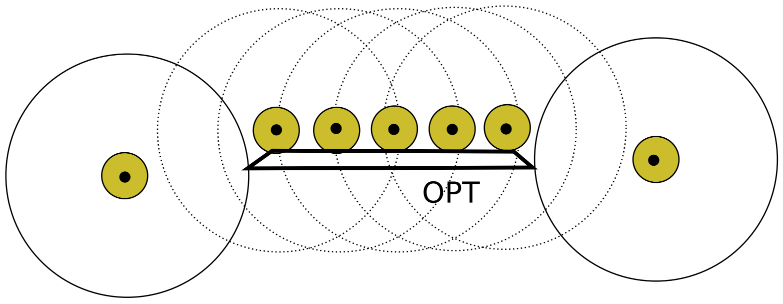


Figure 4.22: A TRT instance. Each sensor has a two ring communication model. If the robot enters the inner disk (shaded region), it downloads data faster than downloading from the outer disk. For this instance the optimal solution visits a mixture of inner and outer disks.

Our Contributions: We first study the general case of collecting data from sensors on the plane. In Section 4.6.3, we present a $(p+q)$-approximation where p and q are the approximation factors for TSPN and k-TSPN problems (These problems will be defined shortly). Next we present simple polynomial algorithms for TRT which have approximation factors $O(T_{out}/T_{in})$ and $O(r_{out}/r_{in})$ (Section 4.6.3 and Section 4.6.3).

We also study the following special cases for TRT problem: non-intersecting outer disks and uniform deployment. When the outer disks are non-intersecting, we present a constant factor approximation algorithm (Section 4.6.4) based on a novel lower bound presented in Theorem 23. When sensors are deployed uniformly (possibly intersecting), we formalize the problem as an integer linear program, solve the relaxed version (linear program) optimally and show how it can be rounded to obtain a 4-approximation (Section 4.6.4). In Section 4.6.5, we present results from simulations and an experiment performed using a ground robot.

4.6.1 Problem Definition

Let $S = \{s_1, \ldots, s_n\}$ be a given set of the locations of n sensors. For each sensor $s \in S$, define D_{in} as the *inner disk* (i.e. the disk centered at s with radius r_{in}) and D_{out} as the *outer disk* (with radius r_{out}). The download time inside the inner disk is T_{in} and the download time inside $D_{out} - D_{in}$ is T_{out}. Without loss of generality, we scale the distances so that the robot's maximum velocity is one unit.

The objective is to find a tour which visits either the inner disk or the outer disk of each sensor and minimizes the total time taken to travel and download data from all sensors. We refer to this problem as the *Two-Ring Tour Problem (TRT)*. We also assume that the robot stops first and then downloads the data. Observe that the robot can reduce the total time of the tour by at most half if it downloads while travelling. Therefore, our results yield approximation algorithms for this model (increased by a factor 2) as well.

The TRT problem is a generalization of the Eucledian TSP problem which is an NP-complete problem. This implies that the there is no algorithm which can solve the TRT problem in polynomial time unless $P = NP$. For such problems, approximation algorithms can be used to get an approximate solution. An approximation algorithm is an algorithm which runs in polynomial time on the size of input and guarantees that the solution is close to the optimal value. For minimization problems like TRT, an $\alpha-$approximation algorithm gives a solution which is at most α times of the optimal

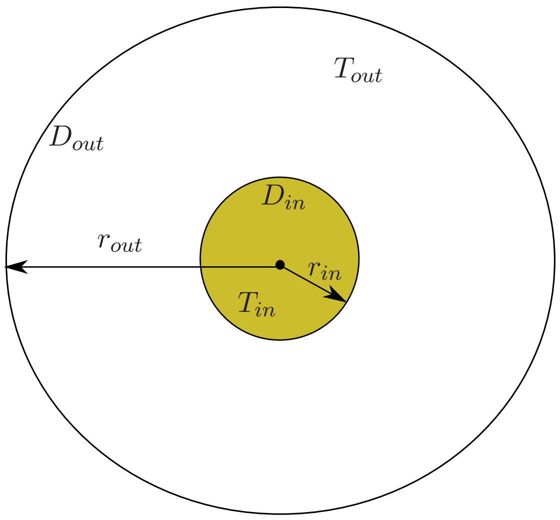


Figure 4.23: The two ring model. Download time is T_{in} in disk D_{in} while it is T_{out} in the region $D_{out} - D_{in}$.

value. In Section 4.6.3 and Section 4.6.4, we present approximation algorithms for the general TRT problem and for some special cases of it.

4.6.2 Structural Properties

In this section, we present some basic properties of an optimal TRT solution.

Proposition 20. *$|C_{in}| \geq |C^*_{out}|$, where $|C_{in}|$ is any tour that visits all inner disks, and C^*_{out} is the optimal TSPN tour visiting outer disks.*

Proposition 20 follows from the fact that any tour that visits the inner disks also visits the outer disks. Using Proposition 20 we obtain a lower bound on the cost of the optimal solution to the TRT problem.

Proposition 21. *$OPT \geq |C^*_{out}| + nT_{in}$, where OPT is the cost incurred by the optimal TRT tour.*

The first term on the right side of the inequality in Proposition 21 is the lower bound on the travel time taken by the optimal tour that visits all the sensors, while the second term is the lower bound on the time to download data from all the sensors.

The following theorem yields a lower bound on the optimal solution. We will use this lower bound in some of the approximation algorithms presented in this section.

Lemma 22. *For any three non-overlapping, equal-size disks on the plane, the length of any path that visits all three disks is lower bounded by αr, where r is the radius of the disks and $\alpha = 0.4786$.*

Proof. Let τ^* be the optimal TSPN solution visiting $n > 2$ disks. Let D_1, D_2 and D_3 be three consecutive disjoint disks on τ^* with centers c_1, c_2 and c_3 respectively. Consider the segment of τ^* which visits these disks. τ^* either touches the boundary of a disk or it crosses the boundary twice. For each disk D_i, we identify a point t_i which is either the touching point or one of the points where τ^* crosses D_i as follows (see Figure 4.24):

- If τ^* touches D_2, pick the touching point as t_2, otherwise pick one of the two points on the boundary of D_2 arbitrarily which is crossed by τ^*

- If τ^* touches D_1, pick the touching point as t_1, otherwise pick the closest crossing point to t_2 as t_1 (similarly choose t_3)

We will present a couple of transformations on the disks such that the length of τ^* will not increase. Afterwards, we will present a lower bound on $|t_1\, t_2| + |t_2\, t_3|$.

In the first transformation, we will replace points t_1 and t_3. In the second transformation, we will move disks D_1 and D_3 in such a way that D_2 touches both disks. Both of these transformations will be done without increasing the total distance. Finally, we will establish the lower bound by optimizing the location of t_2.

Let t'_1 be the point that segment $[t_2\, c_1]$ crosses the boundary of D_1. By moving t_1 to t'_1, we do not increase the total distance (see Figure 4.24). Same observation can be applied for t_3 and t'_3.

If either $|t_1\, t_2|$ or $|t_2\, t_3|$ is greater than or equal to $0.4786r$, lower bound holds since the total distance is at least as claimed. If both distances are less than $0.4786r$ then we do the following transformation. Without loss of generality, let us assume that D_1 is to the left of D_3 (see left of Figure 4.25). If D_1 touches D_2, we do not move D_1. Otherwise, we rotate D_1 along t_2 in counterclockwise direction until D_1 touches D_2. Similarly, if D_3 does not touch D_2, initially, we rotate it in clockwise direction until it touches D_2. Note that this transformation is only a rotation and does not change the total distance. Middle of Figure 4.25 shows this transformation.

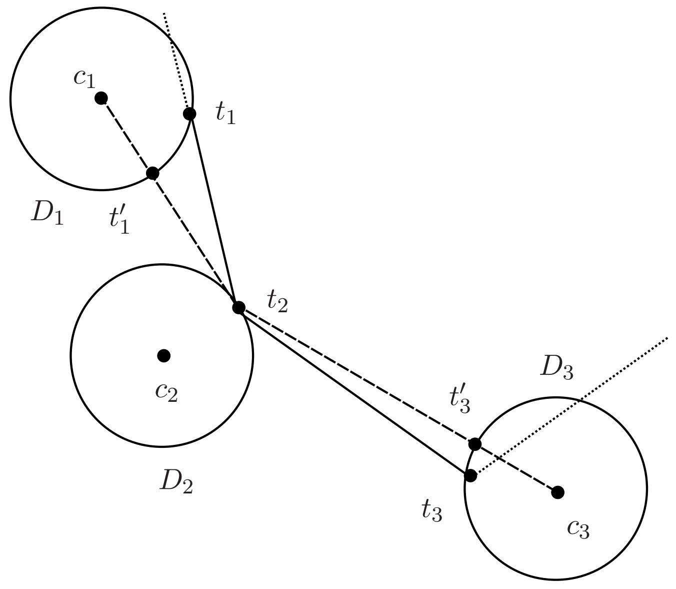


Figure 4.24: Three non-overlapping disks lying in a plane. The part of the optimal TSPN tour t_1, t_2, t_3 which visits disks D_1, D_2 and D_3, respectively. Without increasing the total distance, we can transform t_1 to t'_1 and t_3 to t'_3

In this transformed version of the problem, we formulate the total distance in terms of parameters $\theta = \angle t_2\ c_2\ c'_3$ and $\beta = \angle c'_1\ c_2\ c'_3$ where c'_1 and c'_3 are centers of transformed disks D_1 and D_3. Since t_2 is on the boundary of D_2, we can define all possible locations of t_2 in terms of θ. Since $|t_1\ t_2|$ is less than $0.4786r$, we can show that the angle $\angle c'_1\ t_2\ c_2$ is greater than $\pi/2$. Using the same fact on $|t_2\ t_3|$, we can show that the angle $\angle c'_3\ t_2\ c_2$ is greater than $\pi/2$. Together with the previous inequality, we establish that t_2 is inside the triangle $\triangle c'_1\ c_2\ c'_3$. Using the same distance constraints, we can show that β is upper bounded by $\pi/2$. Moreover, since all disks are non-overlapping, it is lower bounded by $\pi/3$ (the configuration when all disks touch each other). The total distance can be expressed as:

$$r\left(\sqrt{5 - 4\cos(\theta)} + \sqrt{5 - 4\cos(\beta - \theta)} - 2\right)$$

Taking the derivative of this formula with respect to θ and setting it to zero yields that the total distance is minimized when $\theta = \beta/2$ and this value is $2r\left(\sqrt{5 - 4\cos(\beta/2)} - 1\right)$.

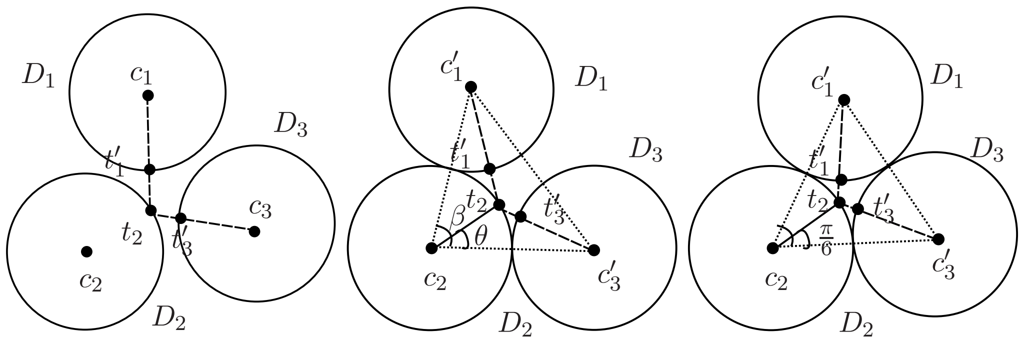


Figure 4.25: **Left:** Initial configurations of circles where $|t_1'\ t_2|, |t_2\ t_3'| < 0.4786r$. **Middle:** After rotation without changing the total distance, D_2 touches both D_1 and D_3. **Right:** The configuration where the total distance is minimum and equals to $0.4786r$.

This value can be further minimized with respect to $\frac{\pi}{3} \leq \beta < \frac{\pi}{2}$ (i.e. $\beta = \pi/3$). Finding this value yields a configuration where all the circles touch each other and t_2 is in the middle of tangent points. In this configuration the total distance is $0.4786 \times r$ which will be used in the next Theorem to find a lower bound on the tour length. This configuration is shown in right of Figure 4.25. $\square$

We use Lemma 22 to find a lower bound on any tour of non-overlapping, equal-sized disks in a plane. This lower bound is used for analysis of algorithms presented in subsequent sections.

Theorem 23. *Any tour τ of n disjoint, equal-sized disks of radius r, satisfies*

$$|\tau| \geq \frac{n}{2}\alpha r, \tag{4.1}$$

where $\alpha = 0.4786$ and $n \geq 3$.

Proof. Take the tour τ. It will give an order of sensors in which to visit them. Let the order be $s_1, s_2, s_3\ , \ldots, s_n$. From Lemma 22 we know that the cost of every sub-path P_i which joins s_i, s_{i+1} and s_{i+2} in τ, is lower bounded by αr for every $i \in 1, n$. Also $|\tau| \geq \frac{1}{2}\sum_{i=1}^{n}|P_i|$. Therefore $|\tau| \geq \frac{n}{2}\alpha r$. $\square$

4.6.3 General Case

In the general case, the communication disks of sensors are placed arbitrarily on the plane possibly overlapping. This section presents three algorithms for this case.

Our first algorithm uses algorithms for TSPN and k-TSPN problems as subroutines. If TSPN solution is p-approximate and k-TSPN solution is q-approximate then our algorithm gives a $(p+q)$-approximate solution for the TRT problem. For example, we can use the PTAS for k-TSPN [113, 117] and PTAS for TSPN from [3] to get a $(2+\epsilon)$-approximate solution for TRT (where ϵ can be made arbitrarily small at the expense of running time).

PTAS algorithms are generally difficult to implement and they have high running times in practice. Therefore, we present two algorithms which are easy to implement with approximation ratios of $O(\frac{T_{out}}{T_{in}})$ and $O(\frac{r_{out}}{r_{in}})$ where $T_{out} > T_{in} > 0$. In practice, the ratios $\frac{T_{out}}{T_{in}}$ and $\frac{r_{out}}{r_{in}}$ are expected to be small, therefore the two algorithms are very relevant for the real world instances of TRT.

General Approximation Algorithm

Let C^* be the optimal tour, S_I be the set of sensors whose inner disks are visited by C^*, and S_O be the remaining sensors whose outer disks are visited by C^*. We have $|S_I| + |S_O| = n$. Then the total cost of this tour is $OPT = |C^*| + |S_I|T_{in} + |S_O|T_{out}$ where T_{in} and T_{out} are the download times from the inner and outer disks of a single sensor.

We observe that C^* is a k-TSPN tour ($k = |S_I|$) of the inner disks and therefore, the optimal k-TSPN tour of all the inner disks is no longer than C^*. Let C_1^* be the optimal k-TSPN tour. By the previous argument, $|C_1^*| \leq |C^*|$. Suppose we can guess k ($|S_I|$), then we find an approximate k-TSPN tour using a q-approximation algorithm for the problem. Then the tour C_1 given by this algorithm will be of length at most $q|C_1^*|$.

Next, let C_2^* be an optimal TSPN tour of the outer disks of sensors not visited by C_1. This tour will be shorter or equal in length than the optimal tour C_{out}^* which visits all the outer disks, i.e., $|C_2^*| \leq |C_{out}^*|$. But the length of the optimal tour of the outer disks is a lower bound on the length C^* (Proposition 20). Therefore, $|C_{out}^*| \leq |C^*|$. We compute a $p-$approximate TSPN tour by using a p-approximation algorithm for the TSPN problem. If C_2 is the tour given by this algorithm, then we have $|C_2| \leq p|C^*|$

Therefore, the total cost incurred by $|C_1 \cup C_2|$ is

$$\begin{aligned} & |C_1| + |C_2| + |S_I|T_{in} + |S_O|T_{out} \\ \leq \; & q|C^*| + p|C^*| + |S_I|T_{in} + |S_O|T_{out} \\ \leq \; & (p+q)OPT \end{aligned}$$

Algorithm 4 implements the steps mentioned above. It guesses k by enumerating all possible k values and then picking up the value of k for which the total cost is minimized.

Algorithm 4 *GENERAL_APPROX_ALGORITHM*

```
$TOUR \leftarrow \phi$
$minCost \leftarrow$ BIGNUMBER
for $k = 1$ to $n$ do
  $C_1 \leftarrow$ $k$-TSPN tour of the inner disks
  $S \leftarrow$ { sensors visited by $C_1$}
  $C_2 \leftarrow$ TSPN tour of the outer disks of the sensors not included in $S$
  $cost \leftarrow |C_1| + |C_2| + kT_{in} + (n-k)T_{out}$
  if $minCost > cost$ then
    $TOUR \leftarrow C_1 \cup C_2$
    $minCost \leftarrow cost$
  end if
end for
return $TOUR$
```

Mitchell presents PTAS algorithms for both TSPN and k-TSPN problems [113]. If we use these algorithms for the case when the outer disks are disjoint, then we get $p = (1+\epsilon/2)$ and $q = (1+\epsilon/2)$. In that case our algorithm yields a $(2+\epsilon)$-approximation factor. In the remaining case, we can use the constant factor approximation algorithm for the outer disks [3] to find a TSPN tour ($p = 11.5$) and the PTAS for inner disks to find a k-TSPN tour which yield to an approximation algorithm with factor $(12.15 + \epsilon)$.

$O(\frac{T_{out}}{T_{in}})$-approximation

The algorithm presented in this section is appropriate for the case when the download times of the inner and the outer disks are comparable. For this scenario, we show that the strategy of visiting just the outer disks of the sensors yields an $O(\frac{T_{out}}{T_{in}})$-approximation for TRT.

First, we use a TSPN algorithm (e.g. [3]) to find a TSPN tour C_{out} of all the outer disks. Let p be the approximation factor for this algorithm and $p \geq 1$. Since the tour visits the outer disks, the robot downloads data from the sensors with the download speed of T_{out}. Therefore, the approximation factor of the algorithm is:

$$\begin{aligned}
\frac{|C_{out}| + nT_{out}}{OPT} &\leq \frac{|C_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \\
&\leq \frac{p|C^*_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \\
&\leq p\frac{|C^*_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \leq p\frac{T_{out}}{T_{in}} \qquad (4.2)
\end{aligned}$$

The first inequality in Equation 4.2 comes from Proposition 21. TSPN approximation directly yields the second inequality. Therefore, our algorithm has an approximation factor of $O(\frac{T_{out}}{T_{in}})$.

$O(\frac{r_{out}}{r_{in}})$-approximation

For some problem instances the ratio of radii may be better than the ratio of download times. In such cases if the ratio of radii is small, we can find an efficient algorithm with the approximation factor of order $O(\frac{r_{out}}{r_{in}})$.

In this algorithm, first we compute a maximal non-overlapping set I of the outer disks. For this we use Algorithm 5. Then we find a TSPN tour C^I_{out} of the disks in I. For each disk $A \in I$, we define the set of sensors whose outer disks intersect with A as S_A. Next, we will show that how we can extend C^I_{out} such that it visits all the inner disks of the sensors in S_A. Also, we assume that $|I| \geq 3$.

Let D be a disk of radius $2r_{out}$ and is co-centered with A. All the sensors in S_A lie on or in D. We traverse D in concentric circles which are distance r_{in} apart as shown in Figure 4.26. This will ensure that all the inner disks of the sensors in S_A are visited.

Algorithm 5 *PARTITION_ALGORITHM(S)*

```
1: $I \leftarrow \phi$
2: while $S \neq \phi$ do
3:     Pick a outer disk $D$ of a sensor $\in S$.
4:     $I \leftarrow I \cup D$
5:     $S_D \leftarrow \{$ sensor with outer disk $D' \in S : D' \cap D \neq \phi\}$
6:     $S \leftarrow S - S_D$
7: end while
8: return MIS set $I$ and all partitions $S_D$.
```

Let d_A be the extra distance traveled in this process and let $k = \lfloor \frac{2r_{out}}{r_{in}} \rfloor$. Then, $d_A = \sum_{i=1}^{k} 2\pi i r_{in} = \pi r_{in} k(k+1) = 2\pi r_{out}(k+1)$. The cost of this tour is $|C_{out}^I| + md_A$ $= |C_{out}^I| + 2m\pi r_{out}(k+1)$, where $m = |I|$. This gives us the approximation ratio:

$$\begin{aligned}
\frac{|C_{out}^I| + md_A + nT_{in}}{|C_{out}{}^*| + nT_{in}} &\leq \frac{|C_{out}^I| + md_A}{|C_{out}{}^*|} \\
&= \frac{|C_{out}^I|}{|C_{out}{}^*|} + \frac{2m\pi r_{out}(k+1)}{|C_{out}^*|} \\
&\leq p + \frac{2m\pi r_{out}(k+1)}{\frac{m}{2} r_{out}\alpha} \\
&= p + \frac{4\pi(k+1)}{\alpha} \qquad (4.3)
\end{aligned}$$

In the second inequality we use p as the approximation ratio for the TSPN algorithm and the lower bound on C_{out}^* is obtained from Theorem 23. Finally, this gives us an $O(\frac{r_{out}}{r_{in}})$ approximation algorithm under the requirement that $|I| \geq 3$.

4.6.4 Special Cases

In this section, we consider efficient solutions for some practical sensor deployments. In the first scenario, we consider a sparse network deployment where communication disks do not overlap. In the second scenario, we consider a common network topology where the sensors are deployed uniformly over a grid.

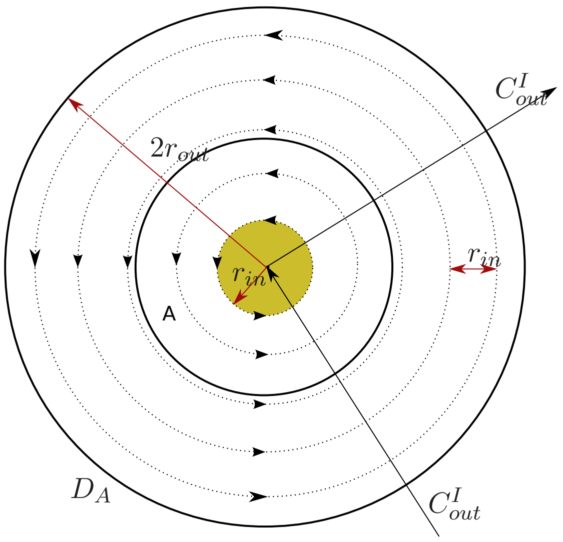


Figure 4.26: In all inner-disk visits, the algorithm chooses to sweep the $2r_{out}$ size disk centered at A in concentric circles which are r_{in} apart.

Non-Overlapping Outer Disks

In this section, we consider the case when all the outer disks are non-overlapping. Our algorithm is simple. We compute a TSPN tour C_{in} of inner disks and download data from inner disks with cost of T_{in} at each disk. This tour can be computed by visiting centers of the disks in polynomial time with $(1+\epsilon)$ approximation using PTAS given in [3].

When $n = 1$, the solution is trivial. For $n = 2$, all possible cases (visiting both inner disks, both outer disks or one inner and one outer disk) can be calculated and the one giving the minimum cost is picked. For $n \geq 3$ we present the following lemma.

Lemma 24. *Given* n *equal size disk of radius* r*, where* $n \geq 3$*, one can compute a TSPN tour* C_{in} *of the inner disks such that*

$$\frac{|C_{in}| + nT_{in}}{|C^*_{out}| + nT_{in}} \leq (1+\epsilon)(1+\frac{4}{\alpha}) \tag{4.4}$$

Proof. We observe that any outer disk TSPN tour can be converted in to an inner disk TSPN tour by extending it at most $2(r_o - r_i)$ in length at each disk. From this observation we get:

$$|C^*_{in}| \leq |C^*_{out}| + 2n(r_o - r_i) \leq |C^*_{out}| + 2nr_o, \tag{4.5}$$

where C^*_{in} is the optimal inner disk TSPN tour. Therefore,

$$\begin{aligned}\frac{|C_{in}| + nT_{in}}{|C^*_{out}| + nT_{in}} &\leq (1+\epsilon)\frac{|C^*_{in}|}{|C^*_{out}|} \leq (1+\epsilon)\left(1 + \frac{2nr_o}{|C^*_{out}|}\right) \\ &\leq (1+\epsilon)(1+\frac{4}{\alpha}) \end{aligned} \tag{4.6}$$

Equation 4.6 is obtained by applying the lower bound obtained in Theorem 23 to Equation 4.5. The $(1+\epsilon)$-approximation is obtained by the PTAS for finding TSPN tour of non-overlapping equal size disks given in [3]. □

This gives us the following result.

Theorem 25. *A TSPN tour of inner disks is a factor $(1+\frac{4}{\alpha})$-approximation for TRT with non-overlapping outer disks ($\alpha = 0.4786$).*

Uniform Deployment

In this section, we consider a common scenario where n^2 sensors are deployed uniformly over an $n \times n$ grid. Let us define a boustrophedon path as a path that goes back and forth along a fixed direction (vertical or horizontal) until it touches the boundary. For the case of uniform deployment, there exists an optimal solution OPT which follows a boustrophedon path. In other words, assuming OPT starts from the top left corner of the grid, first it moves right until it reaches the first vertical line, then follows the vertical line downwards until the bottom of the grid. It then moves right to a vertical line and follows it upwards until it reaches the top and so on. Therefore, if we compute the set of vertical lines traversed by OPT, we can construct the TRT path. Note that this path does not necessarily go through each sensor location (centers of the disks), it simply intersects one of the disks of each sensor.

We restrict the candidate vertical lines to the set of tangent lines L. Each sensor introduces four vertical tangent lines: two tangent to the outer disk and two tangent to the inner disk. It is easy to show that if there exists a solution where there is a vertical line in between two tangent lines in L then we can replace this line with one of its neighbor tangent lines to achieve the same cost. Figure 4.27 shows an instance (only one row) and its vertical tangent lines. From now on, we focus on selecting a subset of L such that the total time is minimized.

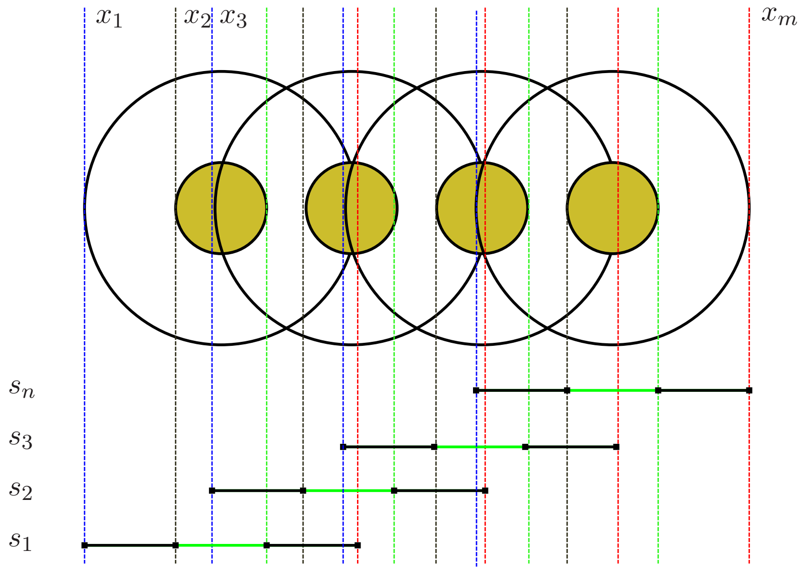


Figure 4.27: Sensors are arranged on an $n \times n$ grid. For each sensor, draw vertical tangents. Two for the outer disk and two for the inner disk. The number of tangents $= m \leq 4n$. The stabbing lines are chosen from this set of tangents.

LP-formulation

For each column i where $1 \leq i \leq n$, we define four binary variables: x_{4i-3} for the left outer tangent line, x_{4i-2} for the left inner tangent line, x_{4i-1} for the right inner tangent line and x_{4i} for the right outer tangent line. We set $x_j = 1$ if and only if the tangent line x_j is traversed. We define a variable y_i such that $y_i = 1$ iff the robot visits the inner disks of sensor column i.

For each column at least one tangent line should be visited hence we have (i) $x_{4i-3} + x_{4i-2} + x_{4i-1} + x_{4i} \geq 1$. Moreover, if one of the inner tangent lines is visited then y_i should be set to 1. We satisfy this by the following two constraints: (ii) $y_i \leq x_{4i-2} + x_{4i-1}$

and (iii) $y_i \leq 1$.

Finally we define the cost of the solution. Let C be the cost of traveling a stabbing line, T_{in} (resp. T_{out}) be the cost of downloading from inner (resp. outer) disks from an entire column.

The total download time is

$$C \sum_{j=1}^{4n} x_j + (T_{in} - T_{out}) \sum_{i=1}^{n} y_i + nT_{out}$$

The integer solution to the above cost function under constraints (i)-(iii) gives us the optimal solution for the uniform case. The following formulation shows the full integer linear program solution:

minimize

$$C \sum_{j=1}^{4n} x_j + (T_{in} - T_{out}) \sum_{i=1}^{n} y_i + nT_{out}$$

such that

$$\begin{aligned} & x_{4i-3} + x_{4i-2} + x_{4i-1} + x_{4i} \geq 1 && 1 \leq i \leq n \\ & y_i \leq x_{4i-2} + x_{4i-1} && 1 \leq i \leq n \\ & y_i \leq 1 && 1 \leq i \leq n \\ & x_i, y_i \in \{0, 1\} \end{aligned}$$

Next we show that the relaxed version can be rounded to obtain a 4-approximation to the uniform case.

Relaxation and Rounding

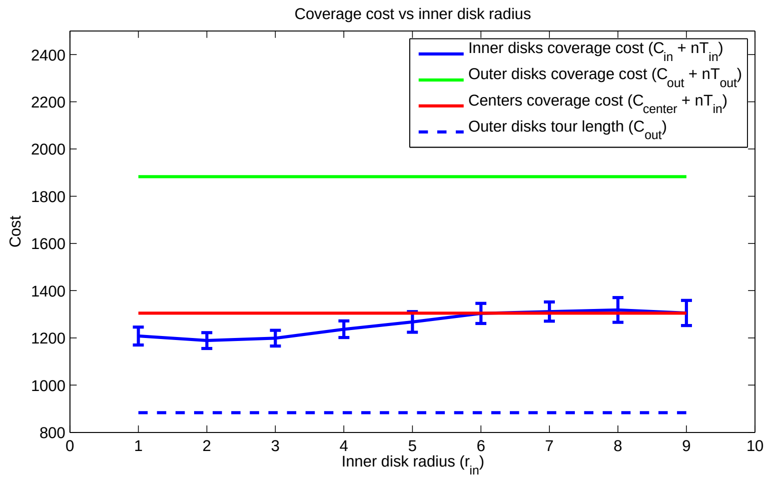
We relax x_j and y_i by replacing the binary constraints with $x_j \geq 0$ and $y_i \geq 0$. Let $OPT(LP)$ be the cost incurred by the optimal solution to the relaxed version. After solving the LP relaxation, we round the solution as follows: If $x_j \geq \frac{1}{4}$ we set it to one. Otherwise, we set it to zero. These are the vertical lines that will be traversed. We then use the values of x_j to determine the values of y_i. Let SOL be the cost incurred by this integer solution, and $OPT(ILP)$ be the cost incurred by the optimal binary solution. Observe that $OPT(LP) \leq OPT(ILP)$ and $SOL \leq 4OPT(LP)$. Therefore, $SOL \leq 4OPT(ILP)$.

We now show that the rounding gives a feasible solution. To see this, observe that due to constraint (i), one of x_{4i-3}, x_{4i-2}, x_{4i-1} and x_{4i} is at least $\frac{1}{4}$. Therefore, after rounding at least one of these stabbing lines is selected, which means that data from every sensor column is collected.

4.6.5 Simulations

In this section we compare the tour of outer disks, the tour of inner disks and the tour of centers of the sensors using simulations. The setup consists of 100 sensors deployed uniformly at random on a 100×100 grid. The radii of the outer disks was set to 10 units for all the trials. Similarly, the download time T_{out} for the outer disks was fixed to 10 units.

First, we varied the inner disk radii r_{in} from 1 to r_{out} and computed the inner disks tour cost. The inner disk download time T_{in} was fixed to 5 units. We performed 100 trials for each r_{in} value and reported the average tour cost. Top:Figure 4.28 shows the average tour cost for different values of the r_{in}. The vertical bars in the figure shows the standard deviation of the tour cost. It was surprising to observe that the inner disk tour's cost increased with increasing r_{in}. On further analysis, we observed that as r_{in} increases the size of the independent set I used to compute TSPN tour decreases (see Bottom:Figure 4.28). When the intersections are few the approximation algorithm yields simple tours and the computed tours are closer to optimal tour. But when the intersections are significant, although the tour cost is bounded by a constant factor, the tours computed are somewhat complex and are relatively less closer to optimal.



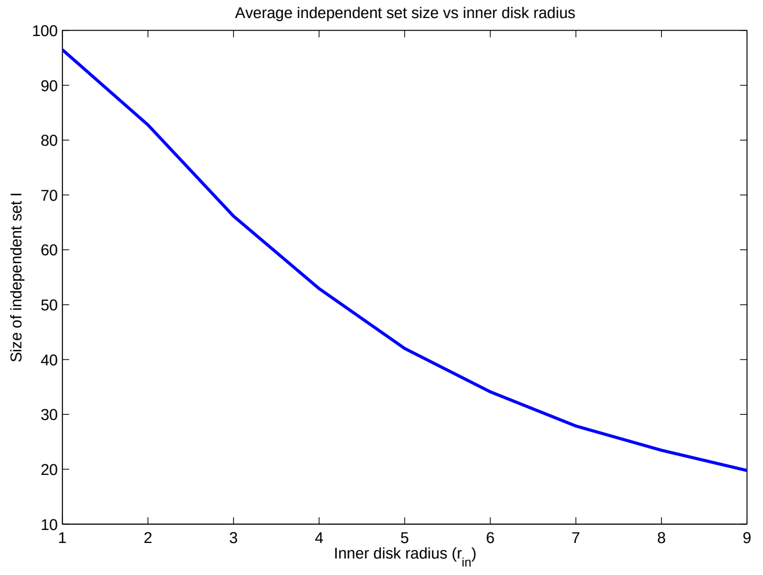


Figure 4.28: **TOP**: This figure shows the change in tour cost with the change in inner disk radius r_{in}. Interestingly the inner disk tour cost increased with increasing r_{in}. **BOTTOM**: This figure shows that the size of independent set I decreases as r_{in} increases. The independent set I is used to compute the TSPN tour using algorithm in [3]. This attributes to the increase in the inner disk tour cost in the TOP figure.

In the second simulation, we vary T_{in} while keeping r_{in} fixed to 5. As expected, the inner disk tour cost varies linearly with T_{in} (Figure 4.29). The dashed line in the figure represents the length of the outer disk tour (it also gives the time to travel since we assume unit speed). From Proposition 20, the length of the optimal outer disk tour is a lower bound on the cost of any inner disk tour. This is observed in the Figure 4.29. When $T_in = 0$, the inner tour cost is slightly lower than the outer tour cost. This is an artifact of the approximation algorithm as described in the previous simulation.

In the final simulation, we fixed both r_{in} and T_{in} to 5. We increased the number of sensors on the grid and computed the tours. Figure 4.30 shows the plot of the experiment. We observe that after approximately 250 sensors, the outer disk tour length does not change significantly. This is because the size of the independent set I used in the approximation algorithm for TSPN [3] does not change with increasing number of sensors. Hence, the tour length also does not vary much. Since the number of sensors increase consequently the download time increases and this causes coverage cost to increase linearly for inner disks tour.

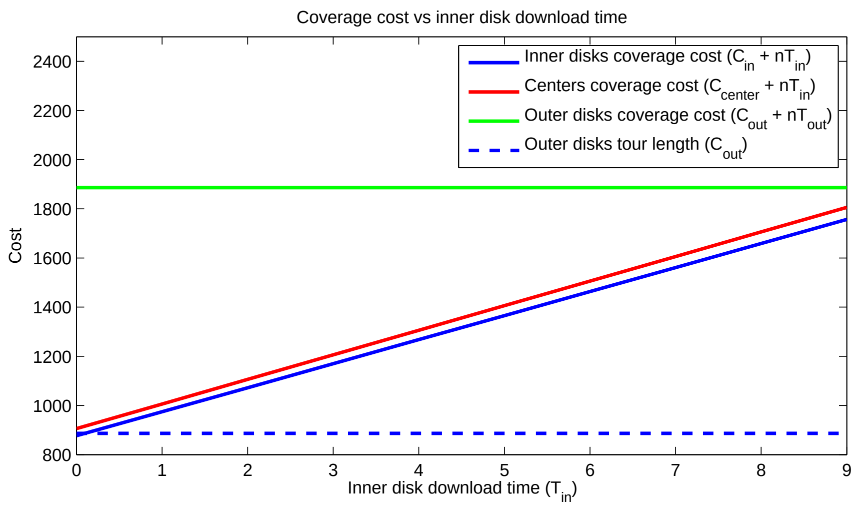


Figure 4.29: The increase in inner disk and center tour cost is linear with increase in T_{in} as expected.

4.6.6 Experiments

In addition to simulations, we performed experiments using wireless sensors (telosB motes) and a custom-built robot. First, we obtained the model parameters. Figure 4.31

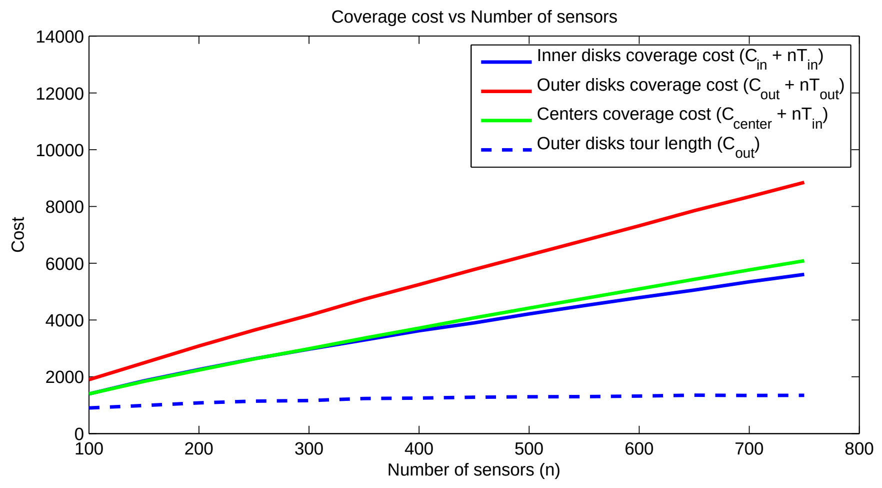


Figure 4.30: As number of sensors are increased tours costs also increase. This is expected as the download time increases linearly with the number of sensors.

shows the average download times as a function of distance from the sensor. For each distance value d, we moved the base mote on a circle which has radius d and centered at the sensor mote location. In each trial, we downloaded 100 packets and recorded their download times. The blue line in the figure shows the average download times and the red error bar shows the minimum and maximum download times observed during each trial. The download times show us that until a distance of 30 feet (gray line) the communication between sensor and base motes is reliable and the download times are short. However beyond 30 feet the communication becomes very unreliable and we observed long download times up to as much as 22 seconds. Beyond 45 feet there was no communication.

Next, we conducted real experiments with a robot to compare the performance of various TRT tours. We used an outdoor robot developed in our lab to collect data from four telosB sensors. These sensors were deployed on the corners of a square field of size 70x70 feet (See Figure 4.32). In this particular experiment, we set the inner disk radius to 18 feet and the outer disk radius to 30 feet.

We tested three natural strategies and compared their performances. In the first strategy the robot followed a TSP tour which visits the centers. In the second and third strategies, the robot visited the outer and inner disks respectively. In each visit the

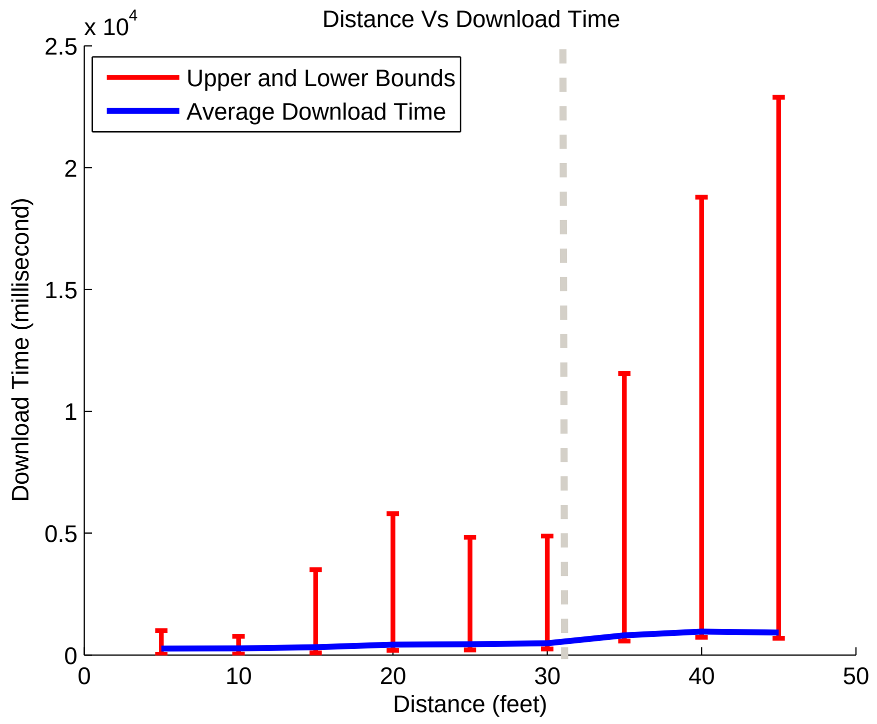
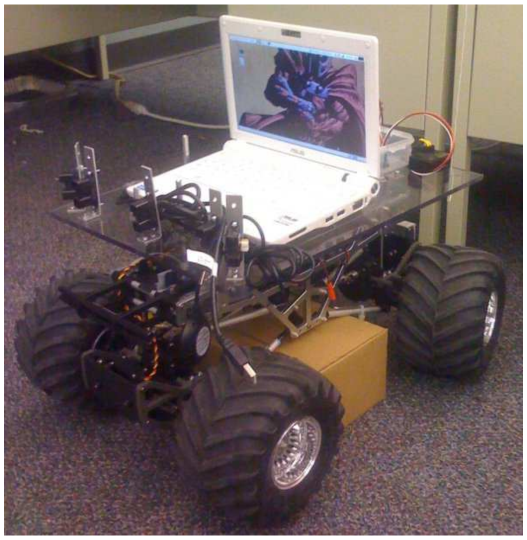


Figure 4.31: Download speed vs distance for communication between two sensor motes.

robot downloaded 100 packets and the time to download varied according to the packet loss.

First we treated the TRT problem as a TSP problem and made the robot visit the sensor locations. In this case the total download time was 8 seconds where as the travel time was 140 seconds (see Figure 4.33 and Table 1). In the second experiment, the robot visited the outer disks which dropped the tour time from 148 seconds to 114 seconds. In this case the download time and the travel time were 50 seconds and 64 seconds respectively. In the last experiment, the robot followed an inner disk tour. The download time drastically decreased from 50 seconds to 9 seconds in this case. However, the travel time increased from 64 seconds to 119 seconds and the overall tour time increased by 14 seconds to 128 seconds.



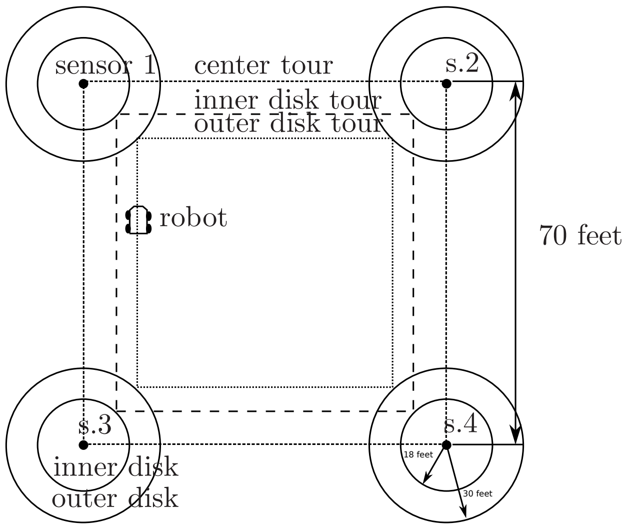


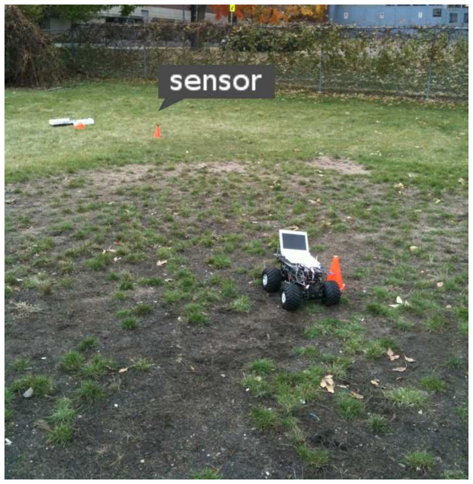
Figure 4.32: **Left:** An outdoor robot developed in our lab was used in the experiments for validating the two-ring model.. **Right:** The setup of the experiment.

	Download	Travel	Total
Visit Disk Centers	8	140	148
Visit Outer Disks	50	64	114
Visit Inner Disks	9	119	128

Table 4.4: Table showing the download and travel times from the three strategies.

4.7 Concluding Remarks

Wireless sensor network technology has the potential to enable major breakthroughs in natural sciences by giving scientists the capability to collect high fidelity data over large geographic regions and extended periods of time. We argue that mobile robots can help sensor networks achieve their full potential. In this chapter, we explored one such synergy between robots and a static sensor field, in which robots act as mules that collect the measurements acquired by the sensors. In Section 4.3, we used a proof-of-concept system to show that this approach is feasible and yields important savings in energy costs, prolonging the network's lifetime.



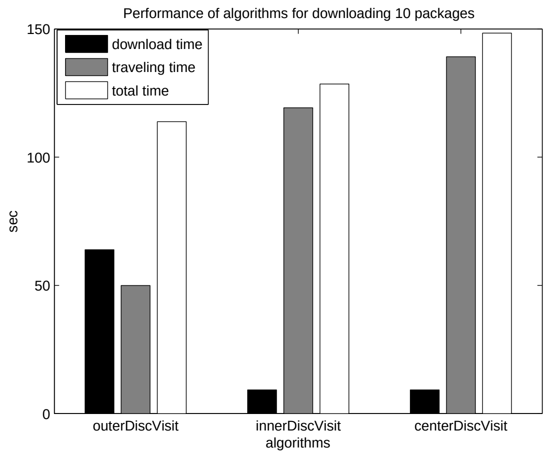


Figure 4.33: **Left:** A snapshot from the experiment. **Right:** Download and travelling time of the following strategies: visit outer disk, visit inner disks and visit disk centers.

In Section 4.4, we investigated further energy improvements by minimizing the number of beacons and reducing the packet loss rate. To minimize the number of retransmissions, we presented a strategy for the robot to adaptively discover a download location where the signal is strong. The strategy is based on insights gathered by empirical results. Finally, we presented an indoor data mule system and showed the improvements in the energy consumption through experiments.

In the rest of the chapter, we studied path planning problem for robots to efficiently gather data from sensors which we call, the Data Gathering Problem (DGP). When the underlying communication model is a disk, we presented an approximation algorithm for DGP in [25]. However, this algorithm ignores the fact that robot can sometimes create a good communication with sensor even when its outside of the communication disk. This might occur due to the navigation uncertainties or constructive multi-path affects. In Section 4.5, we presented a modification to the DGP algorithm which utilizes this fact. In this version of DGP, robot opportunistically downloads data from sensors whenever it hears a good beacon signal.

We also presented the details of an outdoor data mule system. We presented the design choices for an outdoor robot platform and the implementation details for its navigation software. Finally, we presented the results from a field experiment demonstrated the effectiveness and utility of opportunistic DGP algorithm.

In Section 4.6, we introduced a DGP problem where the communication is modeled in terms of two concentric disks around each sensor with different download times. In this formulation, which we call *Two-Ring Tour (TRT)* problem, the mobile entity must decide which disk to visit for each sensor, and spend the corresponding time to download its data.

For the general case, we first presented an algorithm whose approximation is a function of the approximations to k-TSPN and TSPN problems. We also presented two polynomial time algorithms whose performance ratios are proportional to the ratio of the two radii or the ratio of the download times. As demonstrated in an experimental setting, these two parameters are usually small in some practical scenarios.

For two special but common cases of the problem (uniform deployment and sparse deployments where the outer disks are disjoint), we presented constant factor approximation algorithms.

Acknowledgment

The results of this chapter appeared in IEEE Wireless Communications [120], Symposium on Robotics Research [119] and Journal of Field Robotics [25]. This work is joint with Deepak Bhadauria, Nikhil Karnad, Jong Hyun Lim, Prof. Andreas Terzis and Prof. Volkan Isler.

Chapter 5

Multi-Robot Patrol

Consider a scenario where robots (e.g. UAVs) are charged with monitoring a large forest for detecting fires. Due to the limited number of robots and their limited sensing range, it may not be possible to cover every location in the forest at all times. Hence, the robots have to patrol the area efficiently so that a potential fire can be detected as quickly as possible. *Multi-Robot Patrol* (MRP) problem is the problem of finding strategies for a team of robots in order to visit a given set of locations as frequently as possible. An efficient MRP solution can be used in many applications in environmental monitoring (e.g. forest fire detection), surveillance [121] (e.g. detecting intrusions in a border) and search and rescue [122].

A well-known measure for the performance of an MRP solution is its *idleness*. The idleness of a location is defined as the maximum time interval between two consecutive visits to that location [28]. Although this criterion addresses the frequency of visits, it assumes that the locations have the same priority. As shown in [30], in real applications, it is desirable to have a different priority for each location. In the the fire monitoring example, the fire risk might be higher in some locations possibly due to the type of trees or human activity. In such a scenario, higher priority areas should be visited more frequently.

In this work, we introduce a novel version of the MRP problem which captures this property. In the Weighted Multi-Robot Patrol problem (WMRP), the environment is divided into n non-overlapping cells and each cell is assigned a priority (weight) value. We are given m robots which are charged with patrolling the environment. The cost

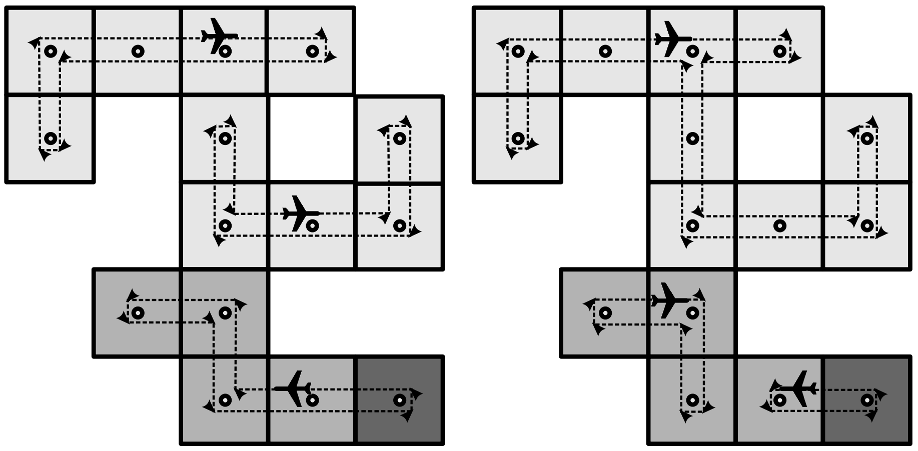


Figure 5.1: An example that demonstrates how priority affects the optimal partitioning. Cells have different priorities represented by their color. Mid-gray and dark-gray colored cells have weights 3 and 5 units, respectively.**Left:** A solution which ignores the priorities would divide the region into three equal parts. **Right:** Optimal solution, when the priority of the cells is incorporated.

incurred by a cell is measured by its *weighted idleness* which is the product of its idleness and weight. The overall cost of a solution is measured by the maximum weighted idleness over all cells. The goal is to partition the environment into m non-overlapping regions (one for each robot) such that the cost is minimized. Having non-overlapping regions is desirable for multi-robot applications (e.g. fleet of UAVs) because the robots can operate independently and collision avoidance is simplified.

In order to demonstrate the practical significance of introducing weights, let us consider an instance with three robots and the simple environment shown in Figure 5.1. The color of a cell represents its priority (weight) (light-gray: 1, mid-gray: 3 and dark-gray: 5 units). For simplicity let us assume that a robot covers a cell when it visits its center. A solution which ignores the priorities would divide the region into three equal parts as shown in Figure 5.1-Left. Assuming the tour length is proportional to the number of cells visited, the cost of this solution is $\max(5 \times 1, 5 \times 1, 5 \times 5) = 25$. This value is computed as follows: The idleness of each cell in a region is computed by the tour length in that region. The maximum weight in the region determines the maximum weighted idleness in that region. The cost of the solution is computed as the maximum weighted idleness over all regions.

When the priority of the cells is incorporated, the cost of the optimal solution is $\max(10 \times 1, 3 \times 3, 2 \times 5) = 10$ (Figure 5.1-Right). This simple example shows how efficient strategies can change with respect to the priorities assigned for each cell. In this chapter, we present a study on how optimal solutions can be computed in the weighted case.

We start the chapter with an overview of related work on MRP and WMRP problems. WMRP problem is formalized in Section 5.2. In Section 5.3, we present an optimal solution to the MRP problem when the topology of the environment can be represented as a tree. An example of such an environment is a simply-connected polygon. The running time of our algorithm for general trees is $O(n^2 \log n \times \log C)$ where n is the number of cells and C is the cost of a single tour that visits every cell in the environment. In some surveillance applications, such as border surveillance, the environment is simpler and can be represented by a path. For such environments, we present an optimal solution in Section 5.4 whose running time is $O(n^2 m)$ where m is the number of robots. This solution removes the dependency on C in the previous solution and provides a better running time.

5.1 Related work

Multi-Robot Patrolling (MRP) problem has recently received the attention of the researchers in both robotics and artificial intelligence communities. This problem was first introduced in [27]. In this paper, the authors define various performance measures for the MRP problem including maximum idleness. Several multi-agent strategies are presented according to various criteria such as agent type, communication model, decision making, etc. In a related work [123], an auction based strategy is presented. Chevaleyre presents the first approximation algorithm for MRP [28]. He presents a *Cyclic Strategy* (CS) where agents are uniformly distributed along a cycle. He proves that the idleness this solution when the cycle is computed by Christofides' TSP algorithm is at most 7 times the idleness of the optimal strategy. A strategy inspired from ant colonies is presented in [124,125]. The strategy is similar to CS except that the cycle is found using a probabilistic ant walk algorithm. The performance of this solution is upper-bounded by a value proportional to the ratio between the maximum and the minimum edge costs in the graph. Santana et. al proposed a reinforcement learning approach for MRP problem [126]. A survey on these results are presented in [127].

Other formulations of the MRP problem were proposed in the literature. Agmon et al. considered the MRP problem in adversarial settings where robots patrol an area while intruders try to penetrate into this area without getting caught [128, 129]. They present strategies which maximize the probability of penetration detection throughout the perimeter of this area. In [130], the authors present an optimal strategy for directional grids where the velocity of robots are constrained depending on the location and the direction of the robots.

All of the previous work listed above assumes that each location has the same priority. A priority based formulation similar to our formulation was first proposed in [29]. In this work, the authors assume that the environment is divided into regions and a graph where each node corresponds to a region is given. For each region there is an idleness constraint and a coverage time. A task strength is computed by a central server which determines the robot strategies using gradient descent without any optimality guarantees. Our work can provide an optimal solution for this problem when the graph is a tree. A reinforcement learning solution is presented in [30] where the reward collected from a node is proportional to the priority and the information gained. Information gain is computed in terms of *exponential idleness* and the uncertainty in the observations. Since the state space is exponential, the authors use several approximations including anytime online algorithms and reactive algorithms to reduce the search space [131]. Hence, their solutions do not guarantee optimality. In [132], the authors present a two-approximation for the problem of minimizing the maximum change of features placed on a compact space. This solution relies on the fact that the locations are placed according to a distribution. In our work, we do not make such assumptions and provide optimal solutions.

5.2 Problem formulation

In this section, we formalize the WMRP problem. We assume that we are given a polygon and m robots. We assume that the polygon is decomposed into n non-overlapping

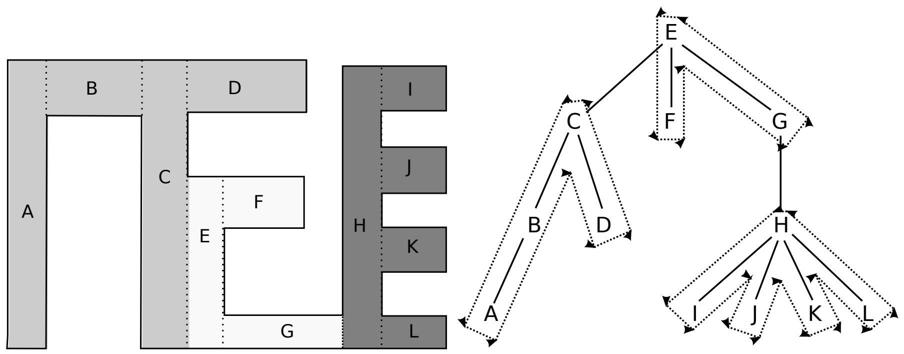


Figure 5.2: The left figure shows a polygon and its decomposition. Each letter represents a cell and each color represents a region. The right figure shows the corresponding regions in the dual-graph (tree) as non-overlapping tours.

cells. A survey on polygon decompositions can be found in [133]. Each cell is associated with a priority (weight) and a coverage cost which is required for a robot to visit each location in the cell. A survey on efficient coverage strategies are presented in [134]. Moreover, we assume that there is a transition cost associated with moving from one cell to the other. Our goal is to partition the environment into m non-overlapping regions. Each robot is assigned to a region and charged with repeatedly visiting the cells in its region. We assume that the coverage costs are much higher than the transition costs. Hence, the robot covers each cell once per tour although it may travel through the cell multiple times. The idleness of the cells inside a region is determined by the time it takes for the robot to cover each cell in its region. Therefore, the idleness of cells inside a region is computed as the sum of the coverage costs and the transitions costs in that region. The objective is to compute the tours of the robots to minimize the overall weighted idleness.

The model described above can be represented as a graph (dual-graph), where each node corresponds to a cell and each edge corresponds to the transition between the two cells. The non-overlapping regions in a polygon can be represented as non-overlapping tours in its dual-graph (see Figure 5.2). Hence, partitioning a polygon into m regions is equivalent to partitioning its dual-graph into m non-overlapping tours. For the rest of the chapter, we focus on finding an optimal partitioning for the dual-graph.

Let $G(V, E)$ be the dual-graph such that $V = \{v_0, v_1, \ldots, v_{n-1}\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges in G. Each edge e_i is associated with an edge length l_i and each node v_i has a weight w_i which represents the priority of that node and c_i which represents the coverage cost. Let m be the number of robots and $\pi_i = \{v_{i0}, v_{i1}, \ldots, v_{in_i}\}$ be a tour that robot i follows. We assume that the robots travel with unit speed. We define the WMRP problem as finding a solution $\Pi = \{\pi_0, \pi_1, \ldots, \pi_{m-1}\}$ which minimizes the cost $C(G, \Pi)$ such that every node is covered (i.e. $\bigcup_{\pi \in \Pi} \pi = V$) and the tours are non-overlapping (i.e. $\forall_{\pi_i \neq \pi_j \in \Pi} \pi_i \cap \pi_j = \emptyset$). Let d_i be the idleness of v_i with strategy Π, we define the cost $C(G, \Pi)$:

$$\max_{v_i \in V} d_i \times w_i = \max_{0 \leq i < m} \left(\|\pi_i\| \max_{v_j \in \pi_i} w_j \right) \tag{5.1}$$

where $\|\pi_i\|$ is the total tour length of robot i. Since robot travels with unit speed, this value is equal to the idleness of each cell covered by robot i. The right equality in Equation 5.1 uses this fact to formalize the cost in terms of maximum of the weights in each partition i.

5.3 WMRP on Tree-like Environments

In this section, we present an optimal WMRP solution for environments whose dual graph is a tree. A target application of this algorithm is simply-connected polygons whose decomposition (e.g. trapezoidal decomposition) can be represented as a tree. Figure 5.2 shows an example simply-connected polygon and its trapezoidal decomposition (left) and its corresponding tree representation (right). Let $T(V, G)$ be a corresponding tree for the given environment, we define two properties that we exploit throughout our algorithm.

Property 1. *Any partitioning of T can be represented by a set of subtrees. An optimal tour on a subtree can be found by traversing each node in a depth-first fashion and crossing each edge twice. The cost (length) of the tour which covers T can be computed using the following formula:*

$$tourCost(T(V, E)) = 2 \sum_{e_i \in E} l_i + \sum_{v_i \in V} c_i \tag{5.2}$$

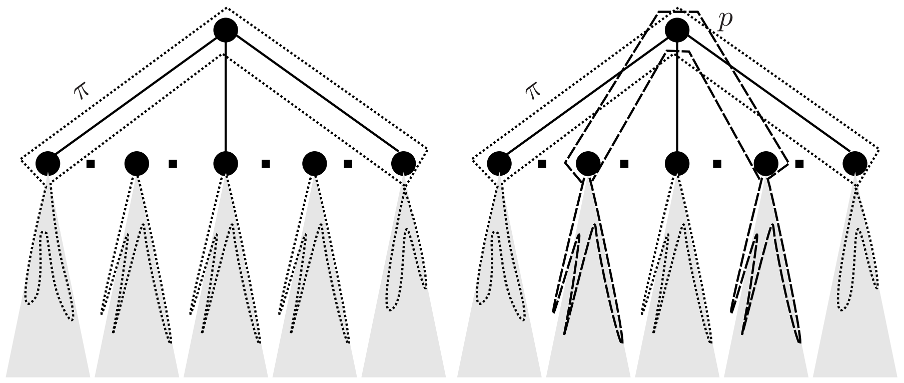


Figure 5.3: Every child of p not covered by the root tour π must be covered by a separate tour as in Figure-Left. Otherwise there is at least two tours overlapping at p as in Figure-Right.

Equation 5.2 sums the transition costs by doubling the edge costs (i.e. l_i) for each edge and the coverage costs (i.e. c_i) for each node in T.

Figure 5.2-Right shows a partitioning of subtrees and their corresponding tours.

Property 2. *Let T_p be a subtree rooted at node p. We call the tour which visits the parent node p as the root tour of T_p. Since tours are non-overlapping, each child node not covered by the root tour π must be covered by a separate tour.*

Figure 5.3-Left shows an example where tours are non-overlapping. Two children not covered by π cannot share the same tour. Otherwise, π and the tour covering these children would overlap on p (Figure 5.3-Right), which would violate the non-overlapping assumption.

5.3.1 Algorithm

Let $C^* = C(T, OPT)$ be the cost incurred by the optimal solution in T with m robots. The algorithm $MinimumCost$ (i.e. Algorithm 6) searches for this value. Given C^* the algorithm *MinimumRobot* (Algorithm 7) computes the minimum number of robots necessary to achieve C^*. The algorithm $MinimumCost$ performs a binary search on possible values of C^* using $MinimumRobot$ and finds the minimum cost C^* that can be achieved by m robots. Algorithm 6 shows how we perform this search. The value ϵ in Algorithm 6 is a precision parameter.

Algorithm 6 *MinimumCost*

Tree: $T(V, E)$, NumOfRobots: m

```
1: $l \leftarrow -\epsilon, u \leftarrow tourCost(T(V, E))$
2: while $l + \epsilon < u$ do
3:   $C \leftarrow (l + u)/2$
4:   $m^* \leftarrow MinimumRobot(T(V, E), C)$
5:   if $m^* > m$ then
6:     $l \leftarrow C$
7:   else
8:     $u \leftarrow C$
9:   end if
10: end while
11: $C^* \leftarrow u$
```

We define $WMRP(C^*)$ as the problem of finding the minimum number of robots necessary to achieve given cost C^*. Since C^* is computed as $C^* = max_{v_i \in V} d_i \times w_i$ the idleness of each node v_i must satisfy the upper bound: $d_i \leq \frac{C^*}{w_i}$. This gives an upper bound on the length of the tour that visits v_i. We define the *tour subtree* $T_p(V_p, E_p)$ as the subtree rooted at p which is constructed from the nodes visited by the root tour. Let π be the root tour and $\|\pi\|$ be its length, the following upper bound must be satisfied: $\|\pi\| \leq \min_{v_i \in V_p} \frac{C^*}{w_i}$.

Now we define the *tour budget*. Let π be the tour on $T_p(V_p, E_p)$, we define the budget b_p as the maximum length of the tour that we can attach to π. This value is given by:

$$b_p = \min_{v_i \in V_p} \frac{C^*}{w_i} - tourCost(T_p(V_p, E_p)) \tag{5.3}$$

In other words, b_p is an upper bound on how much we can extend the tour π without violating the C^* constraint.

There might be multiple optimal solutions to the $WMRP(C^*)$. Let m^* be the minimum number of tours used in the optimal solution to $WMRP(C^*)$. S is the set of optimal solutions such that the number of tours used is m^*. For the rest of the chapter, we will focus on a particular optimal solution in S whose root tour budget is the maximum and we define OPT as this particular solution.

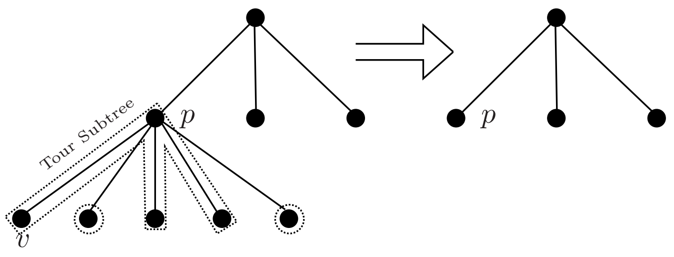


Figure 5.4: Algorithm *MinimumRobot* starts from the deepest node in the tree (i.e. v). Let p be its parent, Algorithm *MaximumSet* finds the tour subtree for the root tour of p. For all children not covered by the root tour we assign a new tour. We remove the children of p and repeat the process.

Now we present the *MinimumRobot* algorithm which is an optimal solution to the $WMRP(C^*)$. Our algorithm starts from the deepest node in the tree. Let p be the parent of this node. We find a root tour which covers as many child nodes as possible while maximizing the tour budget. This tour is found by the algorithm *MaximumSet*. The nodes visited by the root tour are stored as a tree (i.e. tour subtree) so that they can be merged later with the root tours in the upper levels. We remove the children of p and repeat the process until the tree is exhausted. Figure 5.4 illustrates the algorithm *MinimumRobot*.

MinimumRobot takes the cost C^* and the tree $T(V, E)$ as input. For each node v_i, we set the maximum length of the tour that covers v_i, i.e. $d_i = \frac{C^*}{w_i}$ (Line 1). For each node, we set the tour subtree T_i to its root node v_i (Line 2). We start constructing tours beginning from the deepest node in T (Line 5). Let p be the parent of this node, we find the nodes visited by the root tour using the algorithm *MaximumSet* (Line 7). The nodes visited by the root tour is stored in the tour subtree T_p.

Due to Property 2, for each child v_i not covered by the root tour, a separate tour must be created which is performed at Line 8. This tour is constructed from the tour subtree T_i. We remove all children of p from T and repeat this process until T is exhausted (Line 12). Since at least one node is trimmed in each iteration, the algorithm *MinimumRobot* terminates and returns the minimum number of robots necessary to achieve C^* (Line 14).

Algorithm *MaximumSet* takes the parent p and the tour subtree set $\mathcal{T}$ as input.

Algorithm 7 $MinimumRobot$

Tree: $T(V, E)$, Cost: C^*

```
1: $\forall_{v_i \in V} d_i \leftarrow \frac{C^*}{w_i}$
2: $\forall_{i \in V} T_i \leftarrow v_i$, $\mathcal{T} \leftarrow \{T_0, T_1, \ldots, T_{n-1}\}$
3: Tour Index: $i \leftarrow 0$
4: **while** $T \neq \emptyset$ **do**
5:     Find the leaf node $v$ whose depth is the largest
6:     $p \leftarrow parent(v)$
7:     $[T_p] \leftarrow MaximumSet(p, \mathcal{T})$
8:     **for** $v_j \in child(p) \wedge v_j \notin T_p$ **do**
9:         $assign(\pi_i, T_j)$
10:        $i \leftarrow i + 1$
11:    **end for**
12:    $T \leftarrow T - child(p)$
13: **end while**
14: $m \leftarrow i + 1$
```

$MaximumSet$ finds a subset V_{max} from the children of p whose cardinality is maximum. Among all sets with maximum cardinality, it chooses the one with maximum budget. Let P be the set of p and its children and D_i be the set of maximum length constraints in tree T_i. We start with removing the children from P which cannot be covered by the root tour (Line 1). For each node v_i in P, the maximum length of the parent tour is determined by the minimum of maximum tour length constraints in D_i. We find a set V_i of nodes whose minimum maximum tour length constraints are greater than $min(D_i)$ (Line 7). We compute the cost of the root tour which visits each node in V_i. If it is greater than the constraint $min(D_i)$, we remove the node with the highest cost to the tour. Let v_j be a node in V_i, its cost to the root tour is computed as $2l_{jp} + tourCost(T_j)$ where l_{jp} is the edge cost between v_j and p and $tourCost(T_j)$ is the cost of the parent tour of T_j. We repeat this until tour cost satisfies the maximum length constraint (Line 8).

We compute V_i for each node in P and set V_{max} to the set V_i with the maximum cardinality and whose budget is maximum among sets with the same cardinality (Line 12).

Algorithm 8 *MaximumSet*

Parent: p, ConstraintSet: D, SubTreeSet: $\mathcal{T}$

```
1:  $P \leftarrow \{p\} \cup \{i | tourCost(T_p(T_i \cup p)) \leq min(D_i, d_p), v_i \in child(p)\}$
2:  MaxRootSet: $V_{max} \leftarrow \emptyset$
3:  for $v_i \in P$ do
4:      if $min(D_i) > d_p$ then
5:          continue
6:      end if
7:      $V_i \leftarrow \{v_j | min(D_j) \geq min(D_i) \wedge v_j \in P - \{v_i, p\}\}$
8:      while $tourCost(T_p(V_i \cup \{v_i, p\})) > min(D_i)$ do
9:          $removeMaxCost(V_i)$
10:     end while
11:     $V_i \leftarrow V_i \cup \{v_i, p\}$
12:     if $\|V_i\| > \|V_{max}\| \vee$
        $(\|V_i\| = \|V_{max}\| \wedge budget(V_i) > budget(V_{max}))$ then
13:         $V_{max} \leftarrow V_i$
14:     end if
15: end for
16: return $T_p \leftarrow \bigcup_{v_i \in V_{max}} T_i$
```

Algorithm $MaximumSet$ returns the tour subtree T_p constructed from V_{max} (Line 16).

Lemma 26. *Given node p, $MaximumSet$ finds a set of nodes with maximum cardinality whose budget is the maximum among the sets with the same cardinality.*

Proof. Let V^* be the optimal subset and T_i be the subtree with the minimum maximum tour length constraint. Let V_i be the set of nodes whose maximum tour length constraint is not less than $min(D_i)$. Since we compute V_i for all possible nodes, it is one of the sets considered in $MaximumSet$ (Line 1). Let V_i' be the set constructed from the $\|V^*\|$ smallest nodes in V_i. Suppose there exist a node v_j in V_i' but not in V^*. Select the node whose tour cost is the highest in V^* and replace it with v_j. This will lead to a solution whose cardinality is same as V^* but its budget is greater which contradicts with the optimality of V^*. Hence, V_i' must be equal to V^*. Since V_i' is a valid set, it is found by $MaximumSet$ in Line 8 where we remove the costliest nodes from V_i until the budget constraint is satisfied. Hence, $MaximumSet$ returns the optimal subset. $\square$

Let $f < n$ be the branching factor of T, each sorting operation in $MaximumSet$ can be performed in $O(n \log n)$ steps. Each node will be considered at most twice (as a parent or a child) in $MaximumSet$. Hence, the inner loop in $MaximumSet$ is executed at most $2n$ times. The deepest node computation in *MinimumRobot* can be done by traversing T once (i.e. $O(n)$). Therefore, the overall complexity of algorithm *MiniumRobot* is $O(n^2 \log n)$. Since algorithm $MinimumCost$ performs a binary search, $MinimumRobot$ is executed $\log C$ times where $C = tourCost(T(V,E))$. Hence, overall complexity of our algorithm is $O(n^2 \log n \times \log C)$.

5.3.2 Correctness

In this section, we prove that $MinimumRobot$ is an optimal solution for $WMRP(C^*)$. Since the binary search in $MinimumCost$ covers all possible values, the optimality of $MinimumCost$ follows.

Theorem 27. *Given tree $T(V,E)$ and a cost constraint C^*, $MinimumRobot$ finds the minimum number of robots necessary to cover T with cost at most C^*.*

Let $T(p)$ be a tree rooted at p with height $k+1$ and $T(i)$ be a subtree whose root is a child node v_i of p and height is at most k. We define $SOL(T(p))$ as our $MinimumRobot$

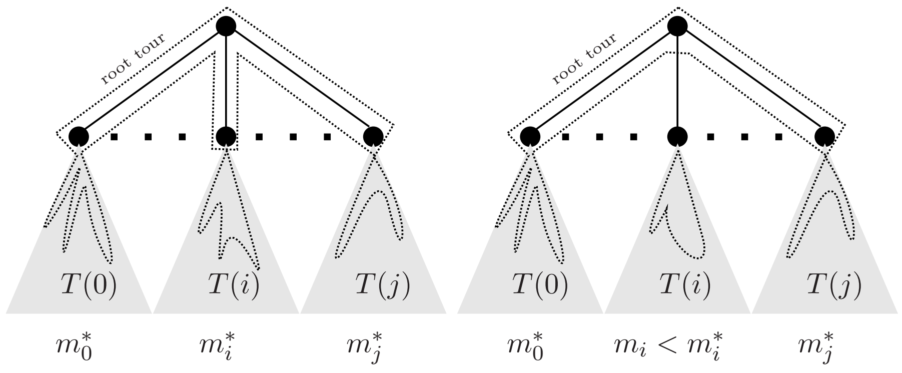


Figure 5.5: Figure-Left shows $OPT(T(p))$. Figure-Right shows $OPT'(T(p))$ which is constructed by cutting the root tour after v_i. We use m_i number of robots to cover $T(i)$ found by $SOL(T(i))$.

solution and $OPT(T(p))$ as the optimal solution on tree $T(p)$. As we mentioned before OPT is a particular optimal solution which uses minimum number of robots and its root budget is maximum. Let m and m^* be the number of tours used by $SOL(T(p))$ and $OPT(T(p))$, respectively. Similarly b and b^* be the root tour budgets. For each subtree $T(i)$ we define corresponding variables: m_i, m_i^*, b_i and b_i^*.

We prove Theorem 27 by induction.

Inductive hypothesis: $SOL(T(p))$ is an optimal solution for all trees of height up to k.

Basis case: When $k = 0$, there exists a single node and it is trivial to show that $SOL(p)$ which assigns a single robot to p is optimal.

Inductive step: Assume that the inductive hypothesis holds for all trees of height up to and including k, we show that $SOL(T(p))$ is optimal for any tree $T(p)$ of height $k+1$. We first show that the number of tours used to cover each subtree $T(i)$ in SOL and OPT satisfy the equality $m_i = m_i^*$.

Lemma 28. $\forall_{v_i \in child(p)} m_i \leq m_i^*$

Lemma 28 follows from the inductive hypothesis.

Lemma 29. $\forall_{v_i \in child(p)} m_i \geq m_i^*$

Proof. Assume that the lemma is not correct. Then there exists a subtree $T(i)$ such that $m_i < m_i^*$. We define a solution $OPT'(T(p))$ which uses $OPT(T(p))$ with a small modification. We cut the root tour after v_i and find a solution for $T(i)$ using SOL. Since $m_i < m_i^*$, the total number of tours used in $OPT'(T(p))$ is not greater than the number of robots used in $OPT(T(p))$. However since the root tour was cut after v_i, the budget of the new root tour is greater. This contradicts with the optimality of $OPT(T(p))$ and the result follows. This construction is shown in Figure 5.5. □

We now have the following lemma which follows from Lemma 28 and Lemma 29.

Lemma 30. $\forall_{v_i \in child(p)} m_i = m_i^*$

We are now ready to prove Theorem 27.

of Theorem 27. From Lemma 30 we know that $\forall_{v_i \in child(p)} m_i = m_i^*$. Moreover from the inductive hypothesis, the budget for each node $\forall_{v_i \in child(p)} b_i \geq b_i^*$ holds. Let π and π^* be the root tour of $SOL(T(p))$ and $OPT(T(p))$, respectively. Similarly let $V_p \subseteq child(p)$ and $V_p^* \subseteq child(p)$ be the set of children of p visited by π and π^*, respectively. For each $v_i \in V_p$ the root tour in T_i is merged with π. Hence, the total number of tours are $m = \sum_{v_i \in child(p)} m_i - \|V_p\| + 1$ and $m^* = \sum_{v_i \in child(p)} m_i^* - \|V_p^*\| + 1$. In Lemma 26, we showed that $MaximumSet$ chooses V_p with the maximum cardinality and maximum budget. Thus, $m \leq m^*$ and $b \geq b^*$ holds which proves the Theorem. □

5.4 WMRP on Path-like Environments

The previous algorithm works for any environment whose dual-graph is a tree. However the time complexity of this algorithm depends on C where C is the time required to covered the environment with single tour. In this section, we remove this dependency for path-like environments. In some surveillance applications such as border surveillance or securing a corridor, the topology is simpler and the dual-graph can be represented as a path. Figure 5.6 shows an example of such an environment.

Let $P(V, E)$ be a path with node set $V = \{v_0, v_1, \dots, v_{n-1}\}$ and edge set $E = \{e_i = (v_i, v_{i+1}),\ i = 0, \dots, n-1\}$. Note that there is a direct correspondence between vertices and edges: edge e_i is the edge between node v_i and v_{i+1}. Given P and the number of

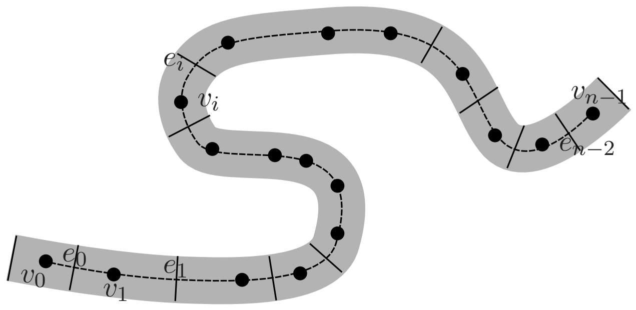


Figure 5.6: An example path-like environment which is observed in some surveillance applications such as border surveillance.

robots m, our goal is to find m tours in P such that the cost measure is minimized. Since tours are non-overlapping, each tour can be represented by its start node $v_i \in P$ and end node $v_j \in P$. Let $\pi(i, j)$ be a tour whose start node and end node are determined by v_i and v_j, then $\pi(i, j) = \{v_i, v_{i+1}, \ldots, v_{j-1}, v_j, v_{j-1}, \ldots, v_{i-2}, v_{i-1}\}$. The cost of π is computed as:

$$c(i, j) = \left(2 \sum_{i \le k < j} l_k + \sum_{i \le k \le j} c_k \right) \times \max_{i \le k \le j} w_k \tag{5.4}$$

The left side of the Equation 5.4 computes the length of the tour and the right side computes the maximum of the node weights.

Our solution for path-like environments is based on dynamic programming. Let $OPT(j, k)$ be the optimal solution which uses k robots to cover nodes 0 through j. For a single tour, we compute $OPT(j, 1)$ as follows: $\forall_{0 \le j < n} OPT(j, 1) = c(0, j)$. Then we find $OPT(j, k)$ using the following recurrence relation:

$$OPT(j, k) = \min_{0 \le i \le j} \max \left(OPT(i-1, k-1), c(i, j) \right) \tag{5.5}$$

We find $OPT(j, k)$ by using the values computed in the previous iterations. For all possible start nodes of the last tour i (i.e. $0 \le i \le j$), we check the cost of the solution whose last tour ends at $i - 1$ using $k - 1$ robots (i.e. $OPT(i - 1, k - 1)$). Overall cost is the max of $OPT(i - 1, k - 1)$ and the cost of the last tour (i.e. $c(i, j)$).

After finding $OPT(n - 1, m)$, we can compute the optimal solution by backtracking from $OPT(n - 1, m)$. For example the last tour can be found by searching for i such that $\max(OPT(i-1, m-1), c(i, n-1)) = OPT(n-1, m)$. Then the last tour starts at i and ends at $n-1$. The rest of the tour can be found using the same procedure.

$OPT(n-1, m)$ can be found in $O(n^2 m)$ time using efficient data structures. We can use a cumulative cost matrix to make $c(i, j)$ queries in constant time. Moreover a pointer to previous entry can be stored to perform backtracking in linear time with m. Finally, since there are $n \times m$ entries and for each entry we need to search over possible start nodes (i.e. at most n possibilities), overall complexity of this algorithm is $O(n^2 m)$.

5.5 Concluding Remarks

In this chapter, we presented a new formulation of the Multi-Robot Patrol (MRP) problem. In this new formulation, an environment is divided into cells and each cell is assigned a priority (weight). In Weighted Multi-Robot Patrol (WMRP) problem, the goal is to find m non-overlapping regions to be covered by m robots independently. For environments whose topology can be represented as a tree (e.g. simply-connected polygons), we presented an optimal solution. The time complexity of this solution is $O(n^2 \log n \times \log C)$ where n is the number of cells and C is the cost of a single tour that visits every cell in the environment. For path-like environments such as a border, we presented an optimal solution whose running time does not depend on C. The time complexity for this solution is $O(n^2 m)$.

The future work includes solving WMRP in general environments such as polygons with obstacles. It is also interesting to relax the non-overlapping constraint so that robots can collaborate to improve the performance of the WMRP.

Chapter 6

Conclusion and Discussion

Recent advances in robotics have made it possible to use mobile robots in complex tasks which were previously not possible. One such task is to provide communication in networked systems. As an example consider the *environmental monitoring* task where scientists collect statistical data using sensors deployed over an environment of interest. The traditional approach to gather data from sensors is to deploy a connected network of wireless sensors so that data can be transfered over the network to a server where scientists store the data. However, the environment of interest could be very large such as an entire forest which makes it infeasible to create a connected network using wireless sensor nodes with limited communication range.

On the other hand, using mobile robots can provide an appealing solution for the data gathering task. Robots which act as *data mules* can autonomously collect data from sensors and carry the data to the server. Alternatively, we can use robots to patrol over the environment to directly collect sensory data. A small delay in data transfer due to data muling is acceptable because scientists are more interested in statistical information then real-time data. Since a single robot can achieve a complex task which otherwise requires many static wireless nodes, using mobility leads to more efficient solutions.

Thanks to recent advances in robotics, now it is feasible to use robots in environmental monitoring tasks. Figure 6.1 and Figure 6.2 show robots that have been used for various environmental monitoring tasks. The robots shown in Figure 6.1 have been developed in our lab (Robotic Sensor Networks Lab at the University of Minnesota).

The first robot (Figure 6.1-Left) is an autonomous land vehicle which was developed for autonomously collecting data from sensors deployed over outdoor environments. A detailed description of this robot was presented in Section 4.5.1. The second robot (Figure 6.1-Right) is a robotic raft. This autonomous robot has been used for tracking carp which were tagged with radio transmitters [4].

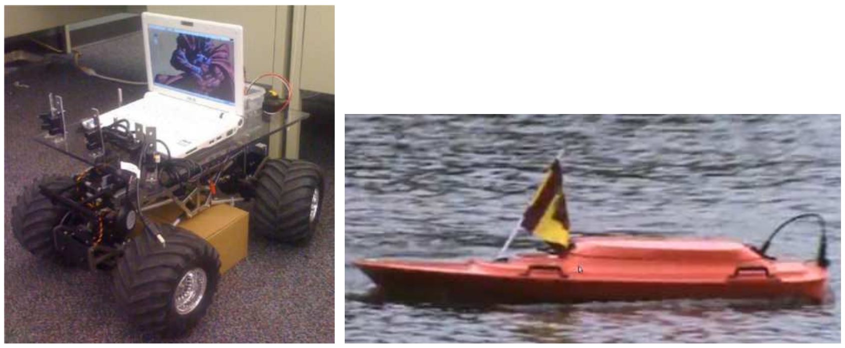


Figure 6.1: **Left:** Cyclops robotic platform developed for data muling. **Right:** Miskin robotic raft developed for tracking carp tagged with radio transmitters [4].

Figure 6.2-Left shows an Autonomous Underwater Vehicle (named Starbug) designed for collecting data from sensors deployed under water [5]. Figure 6.2-Right shows an autonomous snowmobile (named Arctic Crawler) which has been designed for monitoring arctic environments [6]. The last image (Figure 6.2-Bottom) shows an Unmanned Aerial Vehicle (named RCATS/APV-3) which collects vegetation images by flying over vineyards [7].

In this thesis, we focused on theoretical and systems aspect of using robots to improve communication in networked systems. In particular, we studied two types of networked systems and four applications of these systems. In the next section, we present a summary of these results.

6.1 Summary of Our Results

In this dissertation, we studied two types of networked systems. The first system is the *end-to-end network* where we aim to create a persistent connection between two end-points. We study two applications in the first category. In the first application, a mobile

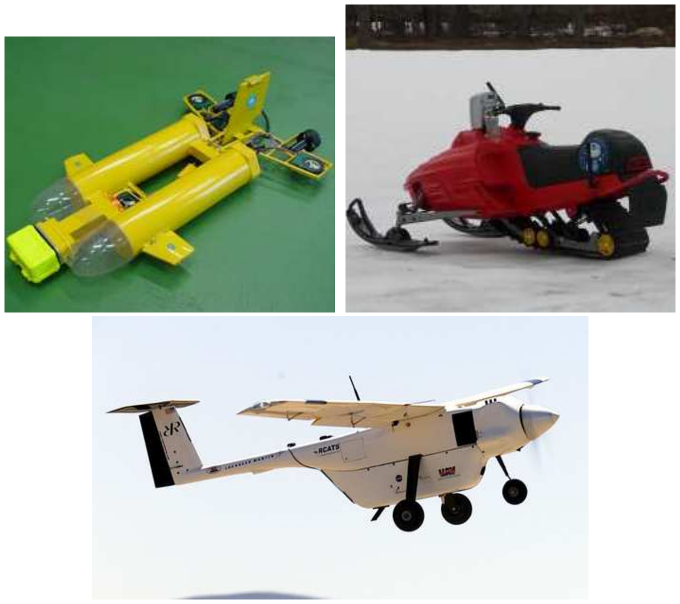


Figure 6.2: **Left:** Starbug robotic platform developed for data muling under water [5]. **Right:** Arctic Crawler robotic snowmobile developed for monitoring arctic environments [6]. **Bottom:** RCATS/APV-3 is a UAV developed for collecting imagery data by flying over vineyards [7].

user requests connectivity with a static base station. We solved this problem by using robots as *robotic routers* to create an adaptive network between the mobile and static end-points. When the user moves, robotic routers autonomously reconfigure themselves to maintain the user's connectivity. In the second application, we used mobile robots to create an on-demand *communication bridge* between two static end-points. In this scenario, we assume robots are scattered around an environment and our goal is to move the robots so that there is a path between the two end-points in the communication graph of the final configuration.

The second system is the *delay-tolerant networks* where a small delay in the data transfer is acceptable. As an example, in *environmental monitoring*, scientists deploy a network of sensors over an environment to collect statistical data. Since statistical information is extracted using data collected for long period of times, it is not necessary to have real-time data. In such applications, we can use mobile robots as *data mules* to autonomously collect data from sensor nodes and carry it to the server where the statistical information is extracted. If the sensors are not sufficient to fully monitor the environment, we can use robots as mobile sensors to directly collect the data from the environment. In this case, multiple robots patrol the environment periodically to minimize the delay between the two consecutive visits to the same location.

Next, we present an overview of our results.

Robotic Routers

In *robotic routers* problem, we find the minimum number of robots and their strategies to create an adaptive network between a mobile user and a static base station. Since the communication model is important to determine connectivity between the robots, our strategies depend on the communication model assumption. In Chapter 2, we present two strategies for the robotic routers problem.

In Section 2.3, we presented optimal solutions that work for arbitrary communication models. We presented two strategies for possible motion models of the user. In the known-user trajectory model, we assume the trajectory of the user is known. This is the case when the user is an entity under our control such as a tele-operated robot. On the other hand, if the user trajectory is unknown, we consider a worst case scenario where an adversarial user tries to break connectivity as quickly as possible. Hence, this strategy works for all possible user trajectories. Although these solutions are optimal in terms of the number of robots used in the system, their running time is exponential in the number of robots. Hence, these solutions might not scale to larger systems.

In our second solution (Section 2.4), we overcame the high time complexity of the previous solution by using a polynomial-time approximate solution. In this solution, we studied a geometric instance of the problem, where the user resides in a geometric environment (e.g. a polygon) and the communication model is determined in terms of geodesic distance. In this model, two nodes are connected iff the length of the shortest path between their locations is not greater than a given threshold. For simply-connected polygons, we presented an optimal solution in terms of the number of robots used in the system. When there is a single obstacle, we presented a strategy which uses at most five times the number of robots used by the optimal solution. We extended this strategy to multiple obstacles by partitioning the environment into convex cells with a single obstacle inside each. Then, we executed the single obstacle strategy inside each partition. Hence, this solution leads to $5h$ approximation where h is the number of obstacles (partitions) in the polygon.

Finally, we discussed the practical utility of using robotic routers with simulations and experiments. We conducted experiments for both known and unknown user trajectory algorithms and reported the results in Section 2.5.

Communication Bridge

In Chapter 3, we introduced the *communication bridge* problem, where the goal is to create a communication bridge between two static end-points while minimizing both the number of robots on the bridge and their maximum (or total) distance traveled. We presented approximation algorithms, for a geometric instance of the problem where robots are restricted to move on the line segment joining the two end-points. For the maximum distance measure, we presented an approximation algorithm whose running time is $O(n^2)$ where n is the number of robots. This solution uses minimum number of robots and the maximum distance travelled is at most $\sqrt{2}$ times the maximum distance travelled in the optimal solution. For the total distance measure, we presented an approximation algorithm whose running time is $O(\frac{n^3 Br}{\epsilon^2})$ where B is the total distance travelled in the optimal solution, r is the communication range and ϵ is the discretization factor. Our solution guarantees that we use the minimum number of robots by a total movement of at most $\sqrt{2}B + n\epsilon$. We also presented an interesting property about the bound on the number of robots. Let $OPT(\infty)$ and $OPT(d)$ be the number of robots used in an unrestricted optimal solution and an optimal solution whose distance constraint is d, respectively. We showed that the constrained version uses at most twice the number of robots used by the unrestricted version.

Data Mules

We have two main contributions in Chapter 4. First, we explored the possible energy savings by using data mules. In Section 4.3, we presented a proof-of-concept indoor data mules system. Moreover, we compared our proposed system with a connected wireless sensor network deployment. In a connected network, data packets are transfered over the network hop-by-hop. Hence, some packets have to be transmitted multiple times. On the other hand, data mules eliminate unnecessary transmissions by directly downloading data from sensors. We compared the energy consumption of both approaches and concluded that the data mules provide an energy efficient solution which reduces the energy consumption and prolongs the life-time of the sensors. In Section 4.4, we achieved more energy savings by using robots to find a good location prior to downloading data. We presented a heuristic for the robot to find a location where the packet loss rate of robot-sensor communication is small. This heuristic is based on insights extracted from our empirical results. In this approach, since sensors are less likely to retransmit lost data packets, we further improved the life-time of sensors.

Our second contribution is on the path-planning problem where our goal is to find paths for multiple robots so that data from all sensors can be gathered as quickly as possible. We considered this data gathering problem under two communication models. In the first model, we assumed the communication model is a disk centered at the sensor location. In our previous work [25], we presented an approximation algorithm for this model. However, this algorithm ignores the fact that the robot can sometimes have good communication with sensor even when its outside of the communication disk. In Section 4.5, we presented a modification of this algorithm where robots opportunistically download data from sensors whenever they detect a good beacon signal. We also presented the details of an outdoor data mule system shown in Figure 6.1-Left. We presented design choices and implementation details of its navigation system. Finally, we presented the practical feasibility of the algorithms and the system through field experiments.

The disk communication model ignores the fact that download time can vary inside the communication range. As proposed in [26], the *two-ring model* is a better model since the time to download data from a sensor s is a function of the locations of the robot and s: If the robot is a distance r_{in} away from s, it can download the sensor's data in T_{in} units of time. If the distance is greater than r_{in} but less than r_{out}, the download time is $T_{out} > T_{in}$. Otherwise, the robot cannot download the data from s. Here, r_{in}, r_{out}, T_{in} and T_{out} are input parameters. In Section 4.6.3, we presented a $(p+q)$-approximation where p and q are the approximation factors for TSPN and k-TSPN problems. In Section 4.6.3 and Section 4.6.3, we presented simple polynomial algorithms that have approximation factors $O(T_{out}/T_{in})$ and $O(r_{out}/r_{in})$. We also studied some special cases. When the outer disks are disjoint, we presented a constant factor approximation (Section 4.6.4). If the sensors deployed uniformly, we presented an integer linear program which yields a 4-approximation (Section 4.6.4). In Section 4.6.5, we presented results from simulations and outdoor experiments.

Multi-Robot Patrol

In Chapter 5, we presented a novel version for the multi-robot patrol problem. In this formulation, we are given a cellular decomposition of the environment. Each cell has a coverage cost for a single robot to patrol the cell. We also have transition costs between cells incurred by the time to travel from one cell to another. Given k robots, our goal is to partition the environment into k non-overlapping partitions such that the maximum cost is minimized. The cost of each partition is the total of the coverage costs and transition costs in that partition. Let G be the dual graph of the decomposition where each vertex corresponds to a cell and each edge corresponds to the transition from the cell to one of its neighbors. If G is a tree, we presented an optimal solution whose running time is $O(n^2 \log n \times \log C)$ where n is the number of cells and C is the cost of a single tour that visits every cell in the environment (Section 5.3). If G is a path, as in Section 5.4, we presented an optimal solution with $O(n^2 m)$ running time where m is the number of robots. This solution removes the dependency on C and, thus provides a better running time.

6.2 Future Research Directions

In this dissertation, we presented several path-planning algorithms to improve communication in networked-systems by using mobile robots. However, there are still many open research problems that need to be addressed. Next, we discuss such future research directions.

Robotic Routers

In Section 2.4, we presented a geometric solution for the robotic routers problem. We presented a constant factor approximation for polygons with a single circular obstacle. Then we extended this to polygons with multiple circular obstacles. We achieved this by partitioning the polygon into cells such that each cell contains a single obstacle. In Section 2.4.4, we used power diagrams to choose the cells that have convex boundaries. Inside each cell, there is a team of robots which executes the single obstacle strategy. We also presented a strategy which guarantees a smooth transition when the user moves from one cell to the other.

The main reason for choosing convex boundaries is to achieve this transition. If the user exits the cell from one point and enters from another point, it is necessary that the shortest path between these points must lie inside the cell. Otherwise, the user can follow the shortest (outer) path while the tip of the connecting arm might have to possibly travel inside because there is an obstacle between the two points. Hence, the connecting arm cannot reach the user when it enters the cell, which would lead to break in connectivity.

In Section 2.4.4, we show how we can extend the power diagrams for polygonal obstacles by using circumcircles. However, if a circumcircle of an obstacle coincides with another obstacle, power diagrams will not work. We can overcome this problem by introducing additional empty convex cells. Figure 6.3-Left shows an example for this partitioning strategy. We introduce an empty triangular cell at the middle of the obstacles so that we can put each obstacle inside a cell with convex boundary. However, to generalize this strategy, we need to show that the number of additional cells is bounded.

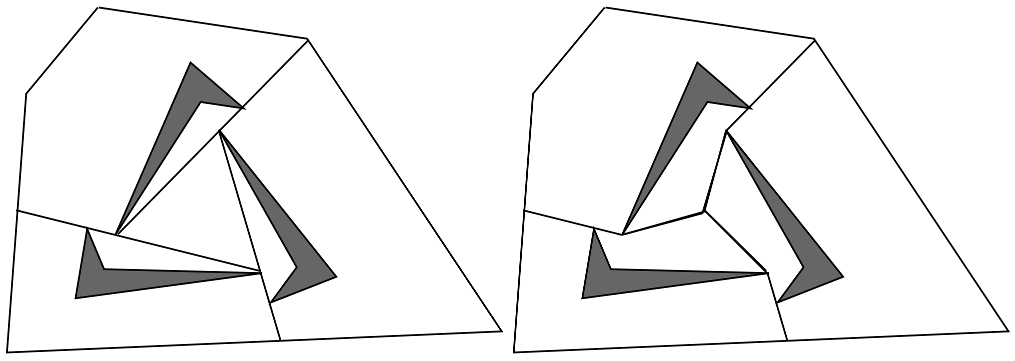


Figure 6.3: **Left:** A convex cell partitioning using an additional empty convex cell. **Rigth:** A geodesic convex partitioning.

While convex boundaries are sufficient, they are not necessary to achieve this property. Rather we can use geodesic convex cells where the shortest path between any two points in the cell lies inside the cell. Figure 6.3-Right, shows an example geodesic convex partitioning. However, finding such partitions remains as a challenging research problem.

Finally determining the computational complexity of this problem is a challenging open question.

Communication Bridge

In the communication bridge problem, we forced the robots to move on the line segment joining two end-points. This solution is practical for line-of-sight communication devices. However, if robots are capable of range communication, then the communication bridge can be any path joining the two end-points. The difficulty in this general formulation of the problem is the lack of an ordering property. Without knowing the order, we cannot determine the possible final locations of the robots. In the line case, we were able to achieve this property by relaxing the problem. Then, we show how much we deviate from the optimal solution due to this relaxation. However, for an arbitrary path, finding an ordering property seems to be a challenging research problem.

Data Mules

In this thesis, we studied the problem of finding efficient paths to collect data from sensors deployed over an environment. We made two assumptions to simplify the problem. First, we assumed that the sensors are deployed over a planar environment. Hence, the paths we compute are straight line segments between the download locations. However, in a real world scenario, there might be obstacles in the environment that overlap with the computed paths. There are several results [135] for computing shortest paths in polygonal environments with obstacles. However, computing data gathering paths for complex environments remains an open research problem.

Another assumption we made is that sensors have enough memory to store collected data between two visits to the sensor. However, some sensors might have limited storage or they might sample more data compared to other sensors. In this case, robots must visit these sensors more frequently than others to prevent possible data loss. Hence, we should incorporate the time window constraint which determines the earliest and latest times to download data. TSP with time windows is a well-known problem [136]. However, since the data gathering task not only includes travel time but also the download time, it is nontrivial to modify existing algorithms for the data gathering problem.

Multi-Robot Patrol

In Weighted Multi-Robot Problem, we presented an optimal strategy for tree-like environments. More specifically, this solution works for a cellular decomposition of an environment whose dual-graph is a tree. In this graph, each vertex corresponds to a cell and each edge corresponds to a neighborhood.

The trapezoidal decomposition is a well-known decomposition method in robotics. The dual graph of the trapezoidal decomposition of an arbitrary polygon is a planar graph. As a special case, the dual-graph of the trapezoidal decomposition of a simply-connected polygon is a tree. Hence, our result works for simply-connected polygons. However, extending it to planar graphs would solve the problem for more practical applications.

Another important research direction is to remove the non-overlapping constraint. We partition the cells into non-overlapping regions to avoid collisions. However, using a collision detection algorithm, one can remove this constraint. Since we might merge partitions that violate the non-overlapping constraint, this would improve the minimum cost. However, computing the optimal solution without this constraint seems to be challenging since we increase the search space of possible solutions.

6.3 Final Remarks

In this dissertation, we studied path-planning problems to improve communication in networked systems using mobile robots. We presented geometric and combinatorial solutions to these problems. The robotics research in this field has been only recently explored. Hence, there are many open research problems. We believe that in the near future, robots will be extensively used in networked systems and will change the way we approach networking tasks. However, to fully utilize mobile robots, it is essential to overcome some real-world challenges. One such challenge is robots' limited energy sources. In this work, we addressed this problem by presenting energy efficient robot strategies. With recent advances in solar power technology, it has become appealing to mount solar panels on robots and make them harvest their own energy. This approach which allows the robots to collect energy along their path introduces many interesting and challenging path-planning problems.

References

[1] iRobot. Landroid, 2011. `http://www.irobot.com/gi/research/Advanced_Platforms/LANdroids_Robot`.

[2] MEMSIC. Telosb datasheet. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152%3Atelosb`.

[3] A. Dumitrescu and J.S.B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

[4] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler. A robotic system for monitoring carp in minnesota lakes. *Journal of Field Robotics*, 27(6):779–789, 2010.

[5] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme. Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3602–3608. IEEE, 2004.

[6] S. Williams, V. Jimenez, L. Antidio, and A.M. Howard. A robotic mobile sensor network for achieving scientific measurements in challenging environments. 2008.

[7] L. Johnson, S. Herwitz, S. Dunagan, B. Lobitz, D. Sullivan, and R. Slye. Collection of ultra high spatial and spectral resolution image data over california vineyards with a small uav. In *Proceedings of the International Symposium on Remote Sensing of Environment*.

[8] G.B. Andeen. *Robot design handbook*. McGraw-Hill Companies, 1988.

[9] S. Squyres. *Roving Mars: Spirit, Opportunity, and the exploration of the red planet*. Hyperion, 2005.

[10] B. Yamauchi. Packbot: A versatile platform for military robotics. In *Proceedings of SPIE*, volume 5422, pages 228–237. Citeseer, 2004.

[11] R. Hirose and T. Takenaka. Development of the humanoid robot asimo. *Honda R&D Technical Review*, 13(1):1–6, 2001.

[12] J.L. Jones. Robots at the tipping point: the road to irobot roomba. *Robotics & Automation Magazine, IEEE*, 13(1):76–78, 2006.

[13] Adept MobileRobots. Research robots, 2011. `http://www.mobilerobots.com/ResearchRobots/ResearchRobots.aspx`.

[14] W. Garage. Personal robot 2 (pr2). *Online: www. willowgarage. com.*

[15] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

[16] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30(2):17, 2009.

[17] R. Musaloiu-E, A. Terzis, K. Szlavecz, A. Szalay, J. Cogan, and J. Gray. Life under your feet: A wireless soil ecology sensor network. In *Proc. 3rd Workshop on Embedded Networked Sensors (EmNets 2006)*. Citeseer, 2006.

[18] R.C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3):215–233, 2003.

[19] O. Tekdas and V. Isler. Robotic routers. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1513–1518. IEEE, 2008.

[20] J. Kutylowski and F. Meyer auf der Heide. Optimal strategies for maintaining a chain of relays between an explorer and a base camp. *Theoretical Computer Science*, 410(36):3391–3405, 2009.

[21] C. Dixon and E.W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. *Mobile Networks and Applications*, 14(3):281–291, 2009.

[22] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1525–1530. IEEE, 2008.

[23] E.D. Demaine, M.T. Hajiaghayi, H. Mahini, A.S. Sayedi-Roshkhar, S. Oveisgharan, and M. Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms (TALG)*, 5(3):1–30, 2009.

[24] A. Kansal, A.A. Somasundara, D.D. Jea, M.B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 111–124, Boston, MA, USA, 2004. ACM.

[25] D. Bhadauria, O. Tekdas, and V. Isler. Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*.

[26] Y. Chen and A. Terzis. Poster abstract: On the spatial characteristics of the gray region for 802.15. 4 radios. In *Information Processing in Sensor Networks, 2009. IPSN 2009. International Conference on*, pages 393–394. IEEE.

[27] A. Machado, G. Ramalho, J.D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. *Multi-Agent-Based Simulation II*, pages 81–97, 2003.

[28] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. 2004.

[29] F. Sempé and A. Drogoul. Adaptive patrol for a group of robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2865–2869. IEEE, 2004.

[30] J.S. Marier, C. Besse, and B. Chaib-Draa. A Markov model for multiagent patrolling in continuous time. In *Neural Information Processing*, pages 648–656. Springer, 2009.

[31] C.E. Perkins. *Ad hoc networking*. Addison-Wesley Professional, 2008.

[32] T. Clausen, G. Hansen, L. Christensen, and G. Behrmann. The optimized link state routing protocol, evaluation through experiments and simulation. In *IEEE Symposium on" Wireless Personal Mobile Communications*. Citeseer, 2001.

[33] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol (OLSR). 2003.

[34] T. Clausen and P. Jacquet. RFC3626: Optimized Link State Routing Protocol (OLSR). *RFC Editor United States*, 2003.

[35] M. Ikeda, G. De Marco, L. Barolli, and M. Takizawa. A BAT in the lab: experimental results of new link state routing protocol. In *22nd International Conference on Advanced Information Networking and Applications*, pages 295–302. IEEE, 2008.

[36] C. Perkins, E. Belding-Royer, S. Das, et al. Ad hoc on-demand distance vector (AODV) routing. 2003.

[37] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *infocom*, page 1405. Published by the IEEE Computer Society, 1997.

[38] V. Park and M.S. Corson. Temporally-ordered routing algorithm (TORA) version 1 functional specification. Technical report, Internet-Draft, draft-ietf-manet-tora-spec-00. txt, 1997.

[39] M.G. Baker, X. Zhao, S. Cheshire, and J. Stone. Supporting mobility in MosquitoNet. In *Proceedings of the 1996 USENIX Technical Conference*, pages 127–140. Citeseer, 1996.

[40] S. Cheshire and M. Baker. A wireless network in mosquitonet. *Micro, IEEE*, 16(1):44–52, 1996.

[41] S. Alpern. The rendezvous search problem. *SIAM Journal on Control and Optimization*, 33:673, 1995.

[42] S. Poduri and G.S. Sukhatme. Latency Analysis of Coalescence for Robot Groups. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3295–3300. IEEE, 2007.

[43] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *Automatic Control, IEEE Transactions on*, 49(4):622–629, 2004.

[44] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme. Robomote: enabling mobility in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 55. IEEE Press, 2005.

[45] A. Kansal, M. Rahimi, D. Estrin, W.J. Kaiser, G.J. Pottie, and M.B. Srivastava. Controlled mobility for sustainable wireless sensor networks. In *Proc. IEEE SECON*, volume 4, 2004.

[46] A. Okabe and A. Suzuki. Locational optimization problems solved through Voronoi diagrams. *European Journal of Operational Research*, 98(3):445–456, 1997.

[47] A. Howard, M.J. Mataric, and G.S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed autonomous robotic systems*, 5:299–308, 2002.

[48] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243–255, 2004.

[49] P. Ogren, E. Fiorelli, and N.E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302, 2004.

[50] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme. Deployment and connectivity repair of a sensor net with a flying robot. *Experimental Robotics IX*, pages 333–343, 2006.

[51] N. Atay and B. Bayazit. Mobile wireless sensor network connectivity repair with k-redundancy. *Algorithmic Foundation of Robotics VIII*, pages 35–49, 2009.

[52] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[53] J. Butterfield, K. Dantu, B. Gerkey, O.C. Jenkins, and G.S. Sukhatme. Autonomous biconnected networks of mobile robots. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, pages 640–646. IEEE, 2008.

[54] D.K. Goldenberg, J. Lin, A.S. Morse, B.E. Rosen, and Y.R. Yang. Towards mobility as a network control primitive. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 163–174. ACM, 2004.

[55] A. Cornejo, F. Kuhn, R. Ley-Wild, and N. Lynch. Keeping mobile robot swarms connected. *Distributed Computing*, pages 496–511, 2009.

[56] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *Automatic Control, IEEE Transactions on*, 51(3):401–420, 2006.

[57] M.M. Zavlanos, A. Jadbabaie, and G.J. Pappas. Flocking while preserving network connectivity. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2919–2924. IEEE, 2007.

[58] M. Lindhé and K.H. Johansson. Communication-aware trajectory tracking. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1519–1524. IEEE, 2008.

[59] Y. Mostofi. Communication-aware motion planning in fading environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3169–3174. IEEE, 2008.

[60] A. Ghaffarkhah and Y. Mostofi. Communication-aware target tracking using navigation functions-centralized case. In *Robot Communication and Coordination,*

2009. ROBOCOMM'09. Second International Conference on, pages 1–8. IEEE, 2009.

[61] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer. Distributed search and rescue with robot and sensor teams. In *Proc. of the 4th Intl. Conf. on Field and Service Robotics, Japan.* Springer, 2003.

[62] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, pages 24–33, 2004.

[63] M.A. Batalin and G.S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2):181–196, 2004.

[64] S.M. LaValle, H.H. González-Banos, C. Becker, and J.C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 731–736. IEEE, 1997.

[65] R. Murrieta-Cid, B. Tovar, and S. Hutchinson. A sampling-based motion planning approach to maintain visibility of unpredictable targets. *Autonomous Robots*, 19(3):285–300, 2005.

[66] S. Bhattacharya, S. Candido, and S. Hutchinson. Motion strategies for surveillance. *Proceedings of Robotics: Science and Systems, Atlanta, GA, USA*, 2007.

[67] F. auf der Heide and B. Schneider. Local strategies for connecting stations by small robotic networks. *Biologically-Inspired Collaborative Computing*, pages 95–104, 2008.

[68] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16:78, 1987.

[69] W. Yang. *An algorithm for network formation and an implementation of a mobile robotic router system.* PhD thesis, Rensselaer Polytechnic Institute, 2008.

[70] O. Tekdas, P.A. Plonski, N. Karnad, and V. Isler. Maintaining connectivity in environments with obstacles. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1952–1957. IEEE, 2010.

[71] O. Tekdas, W. Yang, and V. Isler. Robotic routers: Algorithms and implementation. *The International Journal of Robotics Research*, 29(1):110, 2010.

[72] S. Poduri and G.S. Sukhatme. Achieving connectivity through coalescence in mobile robot networks. In *Proceedings of the 1st international conference on Robot communication and coordination*, pages 1–6. IEEE Press, 2007.

[73] E.J. Anderson and S. Essegaier. Rendezvous search on the line with indistinguishable players. *SIAM Journal on Control and Optimization*, 33:1637, 1995.

[74] Q. Han, D. Du, J. Vera, and L.F. Zuluaga. Improved bounds for the symmetric rendezvous value on the line. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 69–78. Society for Industrial and Applied Mathematics, 2007.

[75] N. Gordon, I.A. Wagner, and A.M. Bruckstein. Gathering multiple robotic a (ge) nts with limited sensing capabilities. *Ant Colony, Optimization and Swarm Intelligence*, pages 60–87, 2004.

[76] E.J. Anderson and RR Weber. The rendezvous problem on discrete locations. *Journal of Applied Probability*, 27(4):839–851, 1990.

[77] V. Baston and S. Gal. Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. *SIAM journal on control and optimization*, 36:1880, 1998.

[78] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[79] E. Meisner, W. Yang, and V. Isler. Probabilistic network formation through coverage and freeze-tag. *Algorithmic Foundation of Robotics VIII*, pages 3–17, 2009.

[80] E.M. Arkin, M.A. Bender, S.P. Fekete, J.S.B. Mitchell, and M. Skutella. The freeze-tag problem: how to wake up a swarm of robots. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 568–577. Society for Industrial and Applied Mathematics, 2002.

[81] E.M. Arkin, M.A. Bender, and D. Ge. Improved approximation algorithms for the freeze-tag problem. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 295–303. ACM, 2003.

[82] O. Tekdas, Y. Kumar, V. Isler, and R. Janardan. Building a communication bridge with mobile hubs. *Algorithmic Aspects of Wireless Sensor Networks*, pages 179–190, 2009.

[83] RS Kanwar, D. Bjorneberg, and D. Baker. An automated system for monitoring the quality and quantity of subsurface drain flow. *Journal of agricultural engineering research*, 73(2):123–129, 1999.

[84] J. Cavender-Bares and NM Holbrook. Hydraulic properties and freezing-induced cavitation in sympatric evergreen and deciduous oaks with contrasting habitats. *Plant, Cell & Environment*, 24(12):1243–1256, 2001.

[85] D.L. Applegate. *The traveling salesman problem: a computational study*. Princeton Univ Pr, 2006.

[86] G.N. Frederickson, M.S. Hecht, and C.E. Kim. Approximation algorithms for some routing problems. In *17th Annual Symposium on Foundations of Computer Science*, pages 216–227. IEEE, 1976.

[87] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107. ACM, 2002.

[88] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 68–77. ACM, 2002.

[89] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, pages 129–145, Palo Alto, CA, USA, 2003. Springer-Verlag.

[90] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. pages 244–257, Marina del Rey, CA, USA, 2005. Springer.

[91] L. Boloni and D. Turgut. Should I send now or send later? A decision-theoretic approach to transmission scheduling in sensor networks with mobile sinks. *Wireless Communications and Mobile Computing*, 8(3):385–403, 2008.

[92] G. Anastasi, M. Conti, E. Monaldi, and A. Passarella. An adaptive data-transfer protocol for sensor networks with Data Mules. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–8, Helsinki, Finland, 2007.

[93] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 1096–1102. IEEE, 2008.

[94] J. Ma, C. Chen, and J.P. Salomaa. mWSN for large scale mobile sensing. *Journal of Signal Processing Systems*, 51(2):195–206, 2008.

[95] E. Ekici, Y. Gu, and D. Bozdag. Mobility-based communication in wireless sensor networks. *IEEE Communications Magazine*, 44(7):56, 2006.

[96] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.G. Lee. Partitioning based mobile element scheduling in wireless sensor networks. In *Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 386–395, Santa Clara, CA, USA, 2005.

[97] J. Xing, H. Wang, K. Han, D.A. Ray, C.H. Huang, L.G. Chemnick, C.B. Stewart, T.R. Disotell, O.A. Ryder, and M.A. Batzer. A mobile element based phylogeny

of Old World monkeys. *Molecular Phylogenetics and Rvolution*, 37(3):872–880, 2005.

[98] A.A. Somasundara, A. Kansal, D.D. Jea, D. Estrin, and M.B. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing*, pages 958–973, 2006.

[99] R. Sugihara and R.K. Gupta. Optimal speed control of mobile node for data collection in sensor networks. *IEEE Transactions on Mobile Computing*, pages 127–139, 2009.

[100] A. Gasparri, B. Krishnamachari, and G.S. Sukhatme. A framework for multi-robot node coverage in sensor networks. *Annals of Mathematics and Artificial Intelligence*, 52(2):281–305, 2008.

[101] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1716–1723. IEEE, 2006.

[102] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Efficient data propagation strategies in wireless sensor networks using a single mobile sink. *Computer Communications*, 31(5):896–914, 2008.

[103] W. Wang, V. Srinivasan, and K.C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 270–283. ACM, 2005.

[104] J. Luo and J.P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1735–1746. IEEE, 2005.

[105] Y. Tirta, Z. Li, Y.H. Lu, and S. Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 515–519. IEEE, 2004.

[106] F.J. Wu, C.F. Huang, and Y.C. Tseng. Data gathering by mobile mules in a spatially separated wireless sensor network. In *International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 293–298, Taipei,Taiwan, 2009. IEEE Computer Society.

[107] B. Yuan, M. Orlowska, and S. Sadiq. On the optimal robot routing problem in wireless sensor networks. *IEEE transactions on knowledge and data engineering*, pages 1252–1261, 2007.

[108] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2091–2098. IEEE, 2006.

[109] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369. IEEE, 2005.

[110] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, et al. Luster: wireless sensor network for environmental research. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 103–116. ACM, 2007.

[111] R. Musaloiu-E, C.J.M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *2008 International Conference on Information Processing in Sensor Networks*, pages 421–432. IEEE, 2008.

[112] N. Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP, 1976.

[113] J.S.B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 11–18. Society for Industrial and Applied Mathematics, 2007.

[114] J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[115] E.M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

[116] K. Elbassioni, A. Fishkin, and R. Sitters. On approximating the TSP with intersecting neighborhoods. *Algorithms and Computation*, pages 213–222, 2006.

[117] J.S.B. Mitchell. Personal Communication.

[118] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde tsp solver, 2006. *http: // www. tsp. gatech. edu/ concorde*.

[119] O. Tekdas, N. Karnad, and V. Isler. Efficient strategies for collecting data from wireless sensor network nodes using mobile robots. In *14th International Symposium on Robotics Research (ISRR)*, 2009.

[120] O. Tekdas, V. Isler, J.H. Lim, and A. Terzis. Using mobile robots to harvest data from sensor fields. *Wireless Communications, IEEE*, 16(1):22–28, 2009.

[121] D. Reis, A. Melo, A. Coelho, and V. Furtado. Towards optimal police patrol routes with genetic algorithms. *Intelligence and Security Informatics*, pages 485–491, 2006.

[122] M. Haque. Sustainable Group Sizes for Multi-Agent Search-and-Patrol Teams. Technical report, Georgia Institute of Technology. School of Electrical and Computer Enginering, 2010.

[123] T. Menezes, P. Tedesco, and G. Ramalho. Negotiator agents for the patrolling task. *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, pages 48–57, 2006.

[124] Y. Elor and A. Bruckstein. Autonomous multi-agent cycle based patrolling. *Swarm Intelligence*, pages 119–130, 2010.

[125] Y. Elor and A.M. Bruckstein. Autonomous Multi-Agent Cycle Based Patrolling. 2009.

[126] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch. Multi-agent patrolling with reinforcement learning. 2004.

[127] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. Recent advances on multi-agent patrolling. *Advances in Artificial Intelligence–SBIA 2004*, pages 126–138, 2004.

[128] N. Agmon, S. Kraus, and G.A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2339–2345. IEEE, 2008.

[129] N. Agmon, S. Kraus, G.A. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1811–1817. Morgan Kaufmann Publishers Inc., 2009.

[130] Y. Elmaliach, N. Agmon, and G.A. Kaminka. Multi-robot area patrol under frequency constraints. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 385–390. IEEE, 2007.

[131] J.S. Marier, C. Besse, and B. Chaib-draa. Solving the continuous time multiagent patrol problem. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 941–946. IEEE, 2010.

[132] S.L. Smith and D. Rus. Multi-robot monitoring in dynamic environments with guaranteed currency of observations. In *IEEE Conf. on Decision and Control, Atlanta, GA*, 2010.

[133] J.M. Keil. Polygon decomposition. *Handbook of Computational Geometry*, pages 491–518, 2000.

[134] H. Choset. Coverage for robotics–A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001.

[135] T. Lozano-Pérez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[136] Y. Dumas, J. Desrosiers, E. Gelinas, and M.M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

Appendix A

Index to Multimedia Extensions

No	Media Type	Description	Link
1	Video	Simulations and videos from proof of concept robotic router experiments (Section 2.5)	`http://rsn.cs.umn.edu/index.php/Robotic_Routers`
2	Video	Videos from proof-of-concept data mule experiments (Section 4.3)	`http://www.youtube.com/watch?v=hEje5oanYC0`
3	Video	Videos from good download location search experiments (Section 4.4)	`http://www.youtube.com/watch?v=21YN4ccz-fM`
4	Video	Videos from outdoor data mule experiments (Section 4.5)	`http://rsn.cs.umn.edu/index.php/Data_Mules`