GeoAI: Challenges and Opportunities

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Yiqun Xie

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

Adviser: Shashi Shekhar

May, 2020

Acknowledgements

I would like to express my deepest gratitude to my advisor Prof. Shashi Shekhar for bringing me into the world of spatial computing and helping me mature along the way. Prof. Shekhar has not only helped me develop my technical skills, but also introduced me to a much more holistic view of research. These far-sighted and well-thought-out philosophies (e.g., the "six questions") have made the greatest impact on the way I perceive and pursue innovative research. I would also like to extend my appreciation to my committee members and collaborators including Prof. Vipin Kumar, Prof. Arindam Banerjee, Prof. Snigdhansu Chatterjee, Prof. Joseph Knight, Prof. David Mulla, Prof. Nicholas Jordan, Prof. Eric Watkins and Mr. Len Kne, for their insightful comments, constructive guidance, inspiring passion and generous support.

I gratefully thank all the members of the spatial computing research group, including Reem Ali, Emre Eftelioglu, Majid Farhadloo, Jayant Gupta, Yan Li, Arun Sharma, Xun Tang, Jiang Zhe, as well as visiting students who helped with my research, including Jiannan Cai, Sam Detor, Ruhi Doshi, Jamal Golmohammadi, Ning Guo and Abigail Roh. They have been an important part of daily life during my Ph.D. study. I also would like to thank group alumni who helped me or gave me constructive suggestions, including Prof. Yan Huang, Dr. Sangho Kim, Prof. Chang-Tien Lu, Dr. Dev Oliver, Dr. Siva Ravada, Prof. Ranga Raju Vatsavai, Prof. Hui Xiong, Prof. KwangSoo Yang, and Prof. Xun Zhou. I also want to thank my other friends in the department (not

listed above), including Rahu Bhojwani, Xuan Bi, Haoji Hu, Xiaowei Jia, Jie Li, Cheng Peng, Gandi Xie, Baoquan Zhang and many more for the fun, help and support during these years. I also thank Kim Koffolt for providing helpful, detailed and systematic suggestions for improving my paper drafts.

Finally, I want to thank my parents and my wife. None of this would have been possible without their love, understanding, help and support.

Dedication

To my parents, Jiangshan Xie and Jianying Zhu.

To my wife, Han Bao.

To all my family and friends.

Abstract

Spatial data have tremendous value and are necessary components in many important societal applications. In recent years, our world has been witnessing a revolution brought by spatial technologies (e.g., Google Maps, Waze, Uber, Lyft, Grubhub, Lime, autonomous driving). According to a McKinsey Global Institute report, location data will generate about $600 billion annual revenue by 2020 with applications in energy, health, retail, etc. The world's economy also heavily relies on location and time data from over 2 billion GPS receivers, and these data are essential to many applications such as banks, airlines, police, emergency services, and telecommunications. Meanwhile, new types of spatial data are emerging at unprecedented scales and varieties (e.g., 25GB/hour per connected vehicle, 47.7PB per year by NASA by 2022).

While spatial data are critical, valuable and collected at massive scales, they pose great challenges to traditional artificial intelligence (AI) techniques when applied to important societal problems. This thesis addresses three of these challenges. First, spatial data (e.g., crime or disease distribution, air quality) are often directly linked to our lived environments. As a result, decisions made on such data tend to have direct impacts on the life of citizens, and thus require statistical robustness to avoid errors which can have high economic and social costs (e.g., false alarm of a crime hotspot). Second, spatial data exhibit interdependency and variability, violating the common i.i.d. (identically and independently distributed) assumption in traditional statistics. This introduces new challenges to traditional optimization problems where spatial interdependency between nearby locations is often neglected and understudied (e.g., spatial contiguity required in land allocation). Finally, data and domain knowledge gaps are common in geospatial problems. For example, while Earth observation imagery is available in the tens

of petabytes, there is very limited training data for many important objects or events (e.g., tree data for preventing fires and power blackouts) and expert knowledge is often required to create such data.

This thesis investigates novel GeoAI techniques to explicitly address these challenges posed by spatial data and problems in three types of AI tasks: learning (i.e., unsupervised clustering); planning (i.e., spatial constraints and optimization); and perception (i.e., geospatial object mapping). First, the thesis proposes a significant DBSCAN approach for statistically-robust clustering to control the rate of spurious patterns. This work introduces a modeling of statistical significance for DBSCAN as well as a dual-convergence algorithm to speed up the computation. Second, the thesis proposes a fragmentation-free spatial allocation algorithm to explicitly model interdependency constraints among decision variables during optimization. Specifically, it introduces an optimization formulation with new spatial decision variables to model spatial contiguity and regularity constraints. It also proposes a hierarchical fragmentation elimination algorithm as well as a multi-layer integral image to efficiently solve the problem in a heuristic manner. Third, the thesis proposes a domain-knowledge assisted learning framework (i.e., TIMBER) to map geospatial objects (i.e., trees) with limited training data. The TIMBER framework introduces a geometric optimization formulation and a fast solver to generate candidates of tree-like structures for the deep learning model, which greatly reduces the difficulty of learning as well as the huge demand on training data. It also proposes a core object reduction algorithm to improve the computational performance. Extensive experiments and case studies show that the proposed approaches greatly outperform existing work in solution quality, and the proposed acceleration techniques greatly reduce the computational cost.

Contents

Acknowledgements **i**

Dedication **iii**

Abstract **iv**

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 GeoAI 1

1.2 Illustrative Application Domain: Smart Cities and Communities 4

1.3 Challenges 6

1.3.1 High Cost of Spurious Results 6

1.3.2 Spatial Interdependency and Variability 7

1.3.3 Spatial Data and Domain Knowledge Gap 8

1.4 Thesis Contributions 9

2 Significant DBSCAN towards Statistically Robust Clustering **12**

2.1 Introduction 12

2.2 Problem Definition 16
2.2.1 Key Concepts 16
2.2.2 Modeling of Statistical Significance 17
2.2.3 Formal Problem Formulation 20
2.3 Significant DBSCAN Clustering 21
2.3.1 Significance Testing for A Single (ϵ, $minPts$) 21
2.3.2 A Dual-Convergence Algorithm 22
2.3.3 Significant Clusters with Varying Densities 33
2.4 Validation 36
2.4.1 Comparative Analysis 36
2.4.2 Sensitivity Analysis 42
2.5 Conclusions and Future Work 45

3 FF-SA: Fragmentation-Free Spatial Allocation 47

3.1 Introduction 47
3.2 Problem Statement: Fragmentation-Free Spatial Allocation 49
3.3 Challenges 52
3.3.1 APX-hardness 52
3.4 Related Work and Limitations 57
3.5 HFE: Hierarchical Fragmentation Eliminator 59
3.5.1 Phase-1: Space Tiling. 60
3.5.2 Phase-2: Choice Assignment 66
3.5.3 Algorithm Acceleration 66
3.5.4 Computational Time Analysis 68
3.6 Validation: Case Study in Agricultural Land Allocation 70
3.6.1 Solution Quality Comparison 72
3.6.2 Execution-Time Analysis 76

3.7 Conclusion and Future Work 78

4 A TIMBER Framework for Mining Urban Tree Inventories Using Remote Sensing Datasets 83

4.1 Introduction 83

4.2 Problem Definition 87

4.2.1 Key Concept 87

4.2.2 Formal Problem Formulation 87

4.3 A TIMBER Framework for Tree Identification 88

4.3.1 Phase 1: Optimization of Tree Locations and Sizes 88

4.3.2 Phase 2: Deep Learning based Urban Tree Filter 94

4.3.3 TIMBER Acceleration 95

4.3.4 Computational Complexity Analysis 97

4.4 Validation 98

4.4.1 Case Study 98

4.4.2 Computational Performance 102

4.5 Conclusions and Future Work 103

5 Conclusions and Future Research Directions 104

5.1 Key Results 104

5.2 Short-Term Future Research Directions 105

5.3 Long-Term Future Research Directions 108

References 111

List of Tables

2.1 Example candidates of test statistics for DBSCAN and their properties . 15

3.1 Benefit and cost values 62

4.1 Summation units in $\boldsymbol{\alpha}$ solutions 96

4.2 Precision of detections in test areas 102

4.3 Recall of detections in test areas 102

4.4 F1 scores of detections in test areas 103

5.1 Taxonomy of thesis contributions and future research directions for short-term (ST, colored in blue) and long-term (LT, colored in green). 106

List of Figures

1.1 Overview of GeoAI. 3

2.1 2000-point data by a homogeneous point process. 19

2.2 Sub-grids of upper and lower bounds. 25

2.3 Progression along trials in dual convergence. 32

2.4 Overall validation framework. 36

2.5 Clustered 10000-point shape data. 39

2.6 Clustered 2000-point test data. 40

2.7 A real-world example: snow emergency tows. 41

2.8 Execution time analysis. 43

3.1 Reduction graph. 52

3.2 Three cases of box-allocation: from MAX-BOX-PACK to FF-SA. (best in color) 56

3.3 A toy example of the spatial allocation problem (best in color). 61

3.4 Region shared by schemes. 64

3.5 Fragmentation elimination. 71

3.6 Spatial allocation result with spatial constraints: $\alpha = 50$ and $\beta = 5$. . . 74

3.7 Spatial allocation result with spatial constraints: $\alpha = 200$ and $\beta = 10$. . 75

3.8 Spatial allocation result with spatial constraints: $\alpha = 800$ and $\beta = 20$. . 76

3.9 Solution quality comparison with 30m-grid. 77

3.10 Solution quality comparison with 60m-grid. 78
3.11 Execution time comparison. 78
4.1 Example of input and output. (best in color) 87
4.2 Comparison of detections. The imagery was collected one year after the NHM, so several trees (removed) may only show up in the NHM. The imagery is only used for visual assistance. (best in color) 100
4.3 Solution quality statistics and execution time. 101

Chapter 1

Introduction

1.1 GeoAI

Geospatial artificial intelligence (GeoAI) is a generalization of traditional artificial intelligence (AI) techniques to address unique challenges posed by geospatial data, i.e., geo-referenced data with location markers. Geospatial data are ubiquitous in many science and application domains such as smart cities, transportation, agriculture, business, public health, public safety, etc. Spatial data come in a large variety of types, ranging from traditional points, polygons, networks and satellite imagery, to more recent types such as geo-tagged social media postings (e.g., tweets), trajectories from smartphones or vehicles (e.g., on-board-diagnostics), aerial imagery from low-flying aircraft or unmanned aerial vehicles, and mixed data from connected vehicles. These data are common, important and emerging at unprecedented scales and varieties. For example, in transportation, each connected vehicle can generate data at 25GB/hour, and the market value of connected vehicles, as predicted by McKinsey, will rise to 170 billion by 2020 [1]. Similarly, Earth observation data are being collected at increasing spatial resolution and frequency because of its value in transportation, infrastructure security,

resource mapping, agriculture monitoring, etc. To date, NASA holds over 27 PB of Earth imagery, and the amount is projected to grow by 47.7 PB/year by 2022 [2].

While spatial data are critical, valuable and collected at massive scales, they pose great challenges (Sec. 1.3) to traditional AI techniques and platforms when applied to important societal problems. GeoAI aims to develop novel techniques to bridge these research gaps. Fig. 1.1 shows an overview of GeoAI in the form of three layers.

The top layer comprises domain sciences that generate and use spatial data as major components of their research methodologies. Examples among many include urban planning (e.g., smart cities [3], resilience [4], equity [5]), transportation (e.g., eco-routing [6, 7, 8]), agriculture (e.g., land and management planning, crop monitoring [12], food-energy-water nexus [13]), and many others.

The bottom layer comprises GeoAI infrastructures, including sensing technologies (e.g., remote sensing satellites, unmanned aerial vehicles, engine sensors on vehicles, smartphones), spatial datasets (e.g., satellite and aerial imagery, LiDAR point cloud, on-board-diagnostic data, check-in data, smartphone trajectories, spatial knowledge graphs [44, 45]), and computing platforms (e.g., local or cloud based processing platforms such as Oracle Spatial [46], high performance computing such as CyberGIS [47] and many discussed in [48], and distributed systems such as SpatialHadoop [49, 50] and GeoSpark [51, 52]).

The middle layer comprises GeoAI techniques, and it is the analytics core and engine of the framework. GeoAI [14] provides a variety of techniques, including learning, planning, perception, etc. In learning, GeoAI provides techniques for both unsupervised learning (e.g., spatial pattern recognition techniques for identifying spatial hotspots and clusters [15, 16, 17, 18, 19], co-locations [21, 22, 23], co-occurrences [6], cascading [24], change footprints [25, 26, 27, 28], spatial outliers [29, 30]) and supervised learning (e.g., classification and regression techniques such as spatial decision tree [31], spatial ensemble

[32] and spatial autoregressive models [33, 34, 35]). For planning tasks, GeoAI provides optimization techniques such as allocation, routing [36, 37, 38], site selection [39, 40], etc. For perception, GeoAI can support geospatial object mapping (e.g., mapping of natural resources, vehicles and buildings [4, 42, 43]), spatial-aware natural language processing, etc.

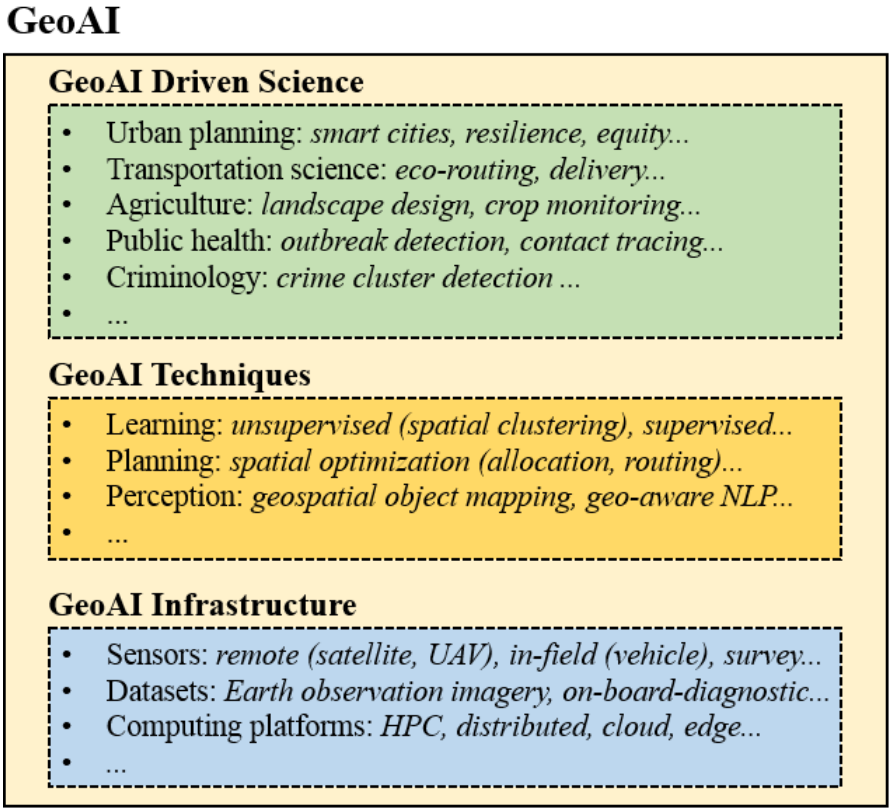


Figure 1.1: Overview of GeoAI.

1.2 Illustrative Application Domain: Smart Cities and Communities

Spatial technologies have transformed our lives with an ever-growing set of tools including Google Maps, Waze, Uber, Lyft, Grubhub, Lime, autonomous driving (e.g., high-definition maps [53]), FarmBeats [54, 55], etc. Increasingly, cities themselves as well as rural areas are also adopting smart spatial technologies.

Smart cities are urban areas that use a wide range of sensors (e.g., smartphones, remote sensing satellites, unmanned aerial vehicles, air quality sensors, traffic cameras) to collect and use data to perform analytics that support smart decision making (e.g., monitoring, operation, management and planning) and promote resilience, public safety, public health, sustainability, green transportation, equity, etc.

With the great variety of spatial data (e.g., imagery, trajectories) being collected at both larger scales and higher resolution (e.g., spatial and temporal), there exists a lot of new opportunities for GeoAI to provide timely solutions to critical societal problems.

One major problem faced by many urban communities is tree management. Increasingly frequent fires and blackouts caused by trees near power lines have raised major safety and resilience concerns in many communities. For example, there have been many historical wildfires caused by untrimmed trees near power lines in recent years, including the Camp fire, which is the largest wildfire in California history. These fires killed many people, destroyed many homes and caused economic losses in the billions [56]. The smoke and unhealthy air also spread to major cities and significantly disturbed daily lives of residents (e.g., many schools were closed [57]). Similarly, in 2018 large-scale blackouts caused by trees falling onto power lines affected millions of people for extended periods at various places including Tallahassee, FL and Cayey, Puerto Rico. To help prevent such disturbances, communities need detailed information about the

locations and types of individual trees relative to city infrastructure.

Tree inventories are also considered vital for dealing with massive insects that threaten various tree species. Emerald ash borer, an invasive species of insect, has spread to over 30 states of the US and many countries around the globe, killing ash trees in millions. In Minnesota, for example, it has affected 21 counties and is projected to cost hundreds of millions if unchecked [58]. Ash trees weakened by the disease pose threats to nearby infrastructures and properties such as buildings, vehicles and power lines. However, due to the large number of ash trees and their wide-distribution, cities are struggling to stay ahead of the rapid spread [58]. As this thesis will show, GeoAI has the potential to address these safety and resilience concerns by automating the mapping of individual trees and their types. Using high resolution satellite and aerial imagery (e.g., 1-meter resolution LiDAR point cloud, 3-inch resolution aerial imagery), communities will be able to identify and target trees that need trimming, removal, etc.

GeoAI can also help mitigate other challenges faced by cities. Spatial hotspot detection and clustering methods [15, 59, 60, 61, 62, 16, 17, 20], for example, have found important urban applications. In public safety, they can help identify neighborhoods with abnormally high rates of crimes. In public health, they can alert policy makers about geographic regions with a potential disease outbreak (e.g., legionnaires' disease, leukemia, cancer) and help them make earlier preparation and intervention. In transportation, hotspots of traffic accidents may reveal potential design issues or unsafe road conditions. These are just a few of many use cases.

Rural communities can also benefit greatly from GeoAI, especially those invested agriculture. The world's population is projected to grow to about 9 billion by 2050, and that requires a 70% increase in food production. However, agriculture, even at today's scale, has already led to severe water quality issues (e.g., the dead zone in the Gulf of Mexico) as a result of fertilizer application. The problem has become so

serious and urgent that Minnesota has enacted a buffer law requiring all farmers to allocate perennial vegetative buffers of up to 50 feet along water bodies [63]. Spatial optimization provides new opportunities to address this conflict by a redesign of the agricultural landscape [64, 9, 10, 11].

1.3 Challenges

Successful application of GeoAI techniques requires considering the unique challenges that working with spatial data poses.

1.3.1 High Cost of Spurious Results

In many societal applications of GeoAI, spurious results carry high economic and social costs. In spatial hotspot detection, for example, incorrectly identifying a community as a crime cluster will lead to lots of unnecessary negative impacts such as reducing property values, hurting local businesses and causing social anxiety. Similarly, a spurious cluster of traffic accidents may waste lots of resources needed to inspect and redesign the roads. In geo-imagery classification, a spurious clear path (e.g., due to uninterpretable classification errors) through the ice floes in the Arctic region may cause a ship to end up stuck in the middle of a long voyage, posing major threats to the safety of crew members and leading to high expenses for rescue. Such social or economic costs cannot be ignored by policy makers. Thus, they often require statistically robust methods or uncertainty-explicit measures to understand and control the rate of spurious results.

1.3.2 Spatial Interdependency and Variability

Spatial interdependency is a fundamental characteristic of spatial data, meaning that their properties (e.g., land use, air quality) at different spatial locations are not independent and are affected by their geographical neighbors. In learning tasks, spatial interdependency manifests as spatial autocorrelation, i.e., the values of data samples (e.g., temperature, air quality) are more similar at nearby locations, violating the common i.i.d. (i.e., independent and identical distribution) assumption underlying many learning models. In spatial optimization, the choices at nearby spatial decision variables need to maintain a certain level of contiguity for practicality concerns. For example, in agricultural landscape design, a land allocation result with many tiny patches of different crops (e.g., a one-square-meter corn field next to a one-square meter soybean field) is not feasible for farmers to implement because they need to use large farm equipment (e.g., large combine harvestors) to manage the fields. In addition, the shape of a patch formed by a homogeneous choice on a group of nearby decision variables needs to be regular (e.g., rectangular) to allow efficient operation of the large equipment.

Spatial variability, a second key property of spatial data, refers to the non-stationary distribution across geographic regions, which also violates the i.i.d. assumption. When spatial variability presents, especially in large-scale applications covering a broad-range of geographic areas, AI techniques that rely on global assumptions about the problem may no longer be appropriate. In spatial pattern mining, a pattern that does not exist at a global scale may exist at a local sub-region, and may be missed by techniques that ignore such variability. In spatial optimization, spatial constraints (e.g., minimum area and shape constraints on farm fields) may vary across geographic regions and need to be handled differently. In supervised learning tasks, non-stationarity in data distributions may require separate training processes to estimate the appropriate parameters for different local geographic areas. Spatial variability can also lead to inconsistent or biased

statistical results summarized using partitions of a geographic space. For example, gerrymandering [65] is a practice used by political parties to gain an unfair advantage during elections by manipulating the boundaries of voting districts.

1.3.3 Spatial Data and Domain Knowledge Gap

AI techniques often rely on high-quality and large-volume datasets to achieve expected performance. However, the availability of such data is very limited in many important spatial applications. Urban tree mapping is such an example. As introduced in Sec. 1.2, the need for inventories of individual trees has become both critical and urgent in many geographic areas. One major challenge of this problem is the lack of ground-truth data (i.e., general and large-scale tree inventories) needed to train deep learning models. In addition, such data are often difficult to collect (e.g., the locations and canopy sizes of trees are hard to measure in the field due to blocked GPS signals) and the generation process requires expert knowledge and software. This significantly limits the usability and value of deep learning based object detection techniques. Similarly, smartphone trajectories are often used in spatial pattern mining (e.g., gathering events, changes in mobility) to support decision-making in smart cities and communities. For example, information about changes in travel patterns is important for policy makers to estimate the effectiveness of social distancing measures during disease outbreaks. However, smartphone trajectories may not be representative of the movement traces of the general public due to factors such as selection bias, and this may cause pattern mining techniques to generate misleading results.

In addition, pure data-driven techniques often do not consider domain knowledge (e.g., laws of physics, epidemiology models), making their results hard-to-interpret and susceptible to domain constraint violations even with large volumes of data. On the

other hand, domain knowledge by itself is also insufficient due to its reliance on simplifying assumptions that may not be well-suited for complex real-world scenarios. This calls for a more holistic view to solve real-world problems by leveraging both data-driven techniques and scientific domain understanding.

1.4 Thesis Contributions

This thesis addresses some of these major challenges by proposing novel GeoAI techniques, including a novel significant DBSCAN for statistically robust clustering with spatial datasets, a fragmentation-free spatial allocation algorithm for land allocation with spatial contiguity and regularity constraints, and a domain-knowledge assisted learning method for mapping geospatial objects (i.e., trees) with limited training datasets. Each chapter is briefly introduced as follows.

- **Chapter 2** discusses a novel statistically-robust spatial clustering algorithm, named Significant DBSCAN, to address the first challenge, i.e., the high cost of spurious patterns, by a new modeling of statistical significance in DBSCAN and a fast dual-convergence algorithm [20]. Given a collection of geo-distributed points, the aim is to detect statistically significant clusters of varying shapes and densities. Spatial clustering has been widely used in public health and safety, transportation, environment studies, etc. The problem is challenging because many application domains have low-tolerance for false positives (e.g., falsely claiming a crime cluster in a community can have serious negative impacts on the residents) and clusters often have irregular shapes. In related work, the spatial scan statistic is a popular technique that can detect significant clusters but it requires clusters to have certain predefined shapes (e.g., circles, rings). In contrast, density-based methods (e.g., DBSCAN) can find clusters of arbitrary shape efficiently but do not consider

statistical significance, making them susceptible to spurious patterns. To address these limitations, I first propose a modeling of statistical significance in DBSCAN based clustering. Then, I propose a baseline Monte Carlo method to estimate the significance of clusters and a dual-convergence algorithm to accelerate the computation. Experiment results show that significant DBSCAN is very effective in removing chance patterns and the dual-convergence algorithm can greatly reduce execution time.

- **Chapter 3** presents a new spatial-dependency-aware optimization algorithm, named Fragmentation-Free Spatial Allocation. This methods addresses the second challenge of spatial interdependency among decision variables with a novel spatial-explicit formulation and a fast hierarchical fragmentation elimination algorithm [9]. Given a grid G containing a set of orthogonal grid cells and a list L of choices on each grid cell (e.g., land use), fragmentation-free spatial allocation (FF-SA) aims to find a tile-partition of G and choice assignment on each tile so that the overall benefit is maximized under both a cost constraint as well as spatial geometric constraints (e.g., minimum tile area, shape). The spatial constraints are necessary to avoid fragmentation and maintain real-world practicality of the spatial allocation result. The application domains include agricultural landscape design (a.k.a., Geodesign), urban land-use planning, building floor zoning, etc. The problem is computationally challenging as an APX-hard problem. Existing spatial allocation techniques either do not consider spatial constraints during space-tiling or are very limited in enumeration space. We propose a Hierarchical Fragmentation Elimination (HFE) algorithm to address the fragmentation issue and significantly increase the enumeration space of tiling schemes. The new algorithm was evaluated through a detailed case study on spatial allocation of agricultural lands in the mid-western US, and the results showed improved solution quality compared

to the existing work.

- **Chapter 4** investigates a new geospatial object mapping algorithm, named TIMBER. It addresses the third challenge of limited and difficult-to-collect training data by assisting deep learning models with domain-theory guided geometric optimization [41, 4]. The TIMBER algorithm focuses specifically on tree mapping because of its societal urgency. Inventories of individual trees are important datasets for applications in urban planning (e.g., removal of ash trees), natural resource management, vegetation disease control, etc. However, tree inventories still remain unavailable in most urban areas due to the difficulty of manual data collection. With the increased availability of high-resolution remote sensing datasets, this work aims to automate tree identification at the individual level in urban areas at a large scale. The problem is challenging due to the complexity of the landscape in urban scenarios (e.g., a mixture of buildings, bridges, trees, etc.) and the lack of ground truth data, which hinders the use of machine learning methods (e.g., convolutional neural networks). This chapter introduces a TIMBER framework to leverage domain knowledge to reduce the need for training data as well as the difficulty of the learning task with geometric-optimization. It also proposes a CORE algorithm to improve the computational efficiency. Experiments show that the proposed framework can significantly improve the accuracy of individual tree detection in urban environments compared to existing approaches, and the acceleration techniques can speed up of tree identification without sacrificing result quality.

- **Chapter 5** concludes the key thesis findings and identifies open directions for future research.

Chapter 2

Significant DBSCAN towards Statistically Robust Clustering

2.1 Introduction

Given a collection of N points in a spatial domain, we aim to detect statistically significant clusters with varying shapes and densities. The points are instances of certain events (e.g., disease, crime).

Detection of significant spatial clusters has been widely applied in important societal domains such as public health, public safety, transportation and environmental science. In public health, epidemiologists use significant clusters (a.k.a, hotspots) to monitor and alert the public about disease outbreaks (e.g., legionnaires' disease, leukemia) [15, 66, 18, 67, 68]. The Research Surveillance Program at the National Cancer Institute has included significant clustering (e.g., SaTScan) as an important methodology and tool [69]. In public safety, police officers use clusters of crime cases to identify neighborhoods with abnormally high crime rates or locate serial criminals [19]. In transportation, many local governments (e.g., US states [70]) have launched "Zero Death" initiatives

to save lives from traffic-related accidents. With significant clustering, planners can find roads with significantly high rates of car accidents or pedestrian fatalities, which indicate potentially unsafe driving conditions (e.g., damaged side walks, pot holes) [71, 72, 73, 74]. In forestry, forest managers use significant clusters to locate high-risk or fragile forest regions and identify potential forest health problems [75, 76]. In environmental science, significant clusters of pollution or contamination can be used to alert administrators of unusual regional phenomenon or problems (e.g., well water contamination by E. coli bacteria) [77, 78]. These are just a few of the many applications using significant clustering.

In these societal applications, decisions often have big impacts, making them have low-tolerance to false positives [14, 79, 17]. For example, false alarms of disease outbreaks may lead to a huge waste of limited public resources (e.g., budgets for sanitation, medication, research) and cause unnecessary social stress or anxiety among lots of people. Similarly, identifying a region as a crime cluster by mistake can greatly reduce the number of people visiting the region, lowering property values and hurting local businesses. For this reason, significance testing is very important and used in many critical societal applications to greatly reduce the risk of false positives.

In related work, the spatial scan statistic [15, 80, 81, 82, 83, 66, 18] is a widely used method for detecting significant clusters. The major strength of the spatial scan statistic is its inclusion of statistical significance. The method introduced a likelihood ratio based framework for testing the significance of spatial clusters and can effectively remove cluster candidates that are likely to be chance patterns. The major limitation of spatial scan statistic based methods is that they require one pre-defined geometric shape (e.g., circle [15, 16, 17, 81], ring [19, 67, 84], rectangle [66, 18], linear[71, 72, 73, 74]), or require certain inputs such as a pre-defined partitioning of the data space (e.g., county boundaries in a state [85]) or specific properties (e.g., event rate per input

partition, attributes for normalization) for enumeration, which are not available in many applications, especially those having data in higher-dimensional spaces. In real-world scenarios, the shapes of clusters are affected by many factors and may change through time, making it difficult to represent them well with pre-defined shapes or partitioning. In addition, the modeling of likelihood ratio in spatial scan statistics did not consider spatial nondeterminism, making it prone to detect very small clusters with just a few points [16, 17, 14]. In the data mining community, DBSCAN [86] is one of the most popular methods for cluster detection, which won the 2014 "Test of Time" Award from ACM SIGKDD. DBSCAN and its variations (e.g., OPTICS [87], H-DBSCAN [88]) are well-known for their ability to detect clusters of varying geometric shape in the presence of noise. It does not require any user input on cluster shapes and can automatically capture them from the data. One major limitation of DBSCAN is that it does not consider the statistical significance of detected clusters [89], leaving space for spurious and chance patterns in the output. While DBSCAN has a default modeling of noise, the type of noise it considers is hard-thresholded and can only handle simple un-clustered points at the per-point level. In a complete random point distribution (i.e., no clustered regions), points that appear near each other just by chance may be treated as valid detections in DBSCAN. This issue was also realized by the creators of the original DBSCAN and OPTICS [88]: "small clusters of objects that may be highly similar to each other just by chance". However, there is still no general modeling of statistical significance for DBSCAN to resolve this problem. A remedy used by some users is to use a threshold of minimum cluster size (e.g., 5 points as defaulted in [90] and many others). While this is an encouraging start, we will show that the absolute definition or value of "small" cannot accurately or appropriately reflect the statistical randomness or significance in the DBSCAN setup (Sec. 2.2.2).

To address these limitations, we aim to join the strengths of statistics and computer

science to improve the flexibility and robustness of spatial clustering. Our contributions are: (1) we explore, propose and compare several modelings of statistical significance in DBSCAN; (2) we focus on a specific modeling and propose a baseline Monte Carlo method to compute the statistical significance; (3) we propose a Dual-Convergence algorithm to improve the computational efficiency of significance testing; and (4) based on the proposed significant DBSCAN (sig-DBSCAN), we present a heuristic search method to describe the detection of clusters with varying densities in the context of significance testing.

Experiments show that the proposed sig-DBSCAN can effectively eliminate spurious patterns with significance testing and the Dual-Convergence algorithm can greatly reduce the computational cost.

Table 2.1: Example candidates of test statistics for DBSCAN and their properties

Properties	**Density** d	**Likelihood ratio** lr	**Cluster size** n
Area of cluster	Required	Required	N/A
Normalization	Area	Area + null hypothesis	fixed radius [16]; $(\epsilon, minPts)$ in DBSCAN
Bias towards small clusters	Yes [91, 16]	Yes (less) [16, 14]	No
Computation	Area dependent	Area dependent	$O(1)$ for a given cluster

2.2 Problem Definition

2.2.1 Key Concepts

Point distribution: A collection of N geo-distributed points of certain events (e.g., disease) in a spatial domain.

Point Process: A statistical process that governs the generation of a point distribution. It determines the probability of having a point at each location within the spatial domain. A homogeneous point process (e.g., complete spatial randomness) has identical probability across all locations (i.e., no true cluster). In contrast, a biased/clustered point process has higher probabilities for locations inside the clustered regions and lower outside. Point process is used to define the null and alternative hypotheses in significance testing.

DBSCAN: Density-Based Spatial Clustering of Applications with Noise [86]. It takes two inputs: (1) search radius ϵ, and (2) minimum number of points $minPts$. With $(\epsilon, minPts)$, DBSCAN classifies a point as a core point if its ϵ neighborhood contains at least $minPts$ points. In short, once a core point c is found, DBSCAN initializes a cluster point set $P(\epsilon)$ of all points with its ϵ neighborhood. For any point $c' \in P(\epsilon)$ that is also a core point, it expands $P(\epsilon)$ by adding all points (without duplication) in the ϵ neighborhood of c' to $P(\epsilon)$. The expansion continues until there is no unexpanded core point left in $P(\epsilon)$, including any newly added ones in the process. All the points in $P(\epsilon)$ form a single cluster. Then DBSCAN continues to find another cluster if it exists. Finally, all points that do not belong to any $P(\epsilon)$ are labeled as noise.

Test statistic: A random variable used to summarize the sample data (e.g., a cluster in a point distribution) and test the hypotheses. In this context, it can be considered as a score calculated from the data (e.g., density of a cluster in a point distribution). The significance of the score determines whether to reject the null hypothesis.

2.2.2 Modeling of Statistical Significance

We explore and compare several different modelings of statistical significance for DBSCAN clustering. To be specific, the significance of DBSCAN we are modeling here is in the context of a single $(\epsilon, minPts)$ combination given by a user. The use of this modeling in the context of varying $(\epsilon, minPts)$ will be discussed in Sec. 2.3.3.

The null hypothesis states that a cluster is generated by a homogeneous point process. In contrast, the alternative hypothesis states that it is generated by a clustered/biased point process (Sec. 2.2.1). Note that for significant cluster detection, it is insufficient to just check whether the entire point distribution belongs to a homogeneous point process or not (e.g., using Ripley's K function). For example, in a point distribution generated by a clustered point process, a clustering method such as DBSCAN often identifies a mixture of significant clusters and chance patterns. As a result, just knowing that the overall point distribution is clustered cannot help us filter out the chance patterns. This is why the hypotheses need to be fine-grained to cluster levels.

We explore several test statistics for the significance or hypothesis testing as shown in Table 2.1. For a given cluster from a point distribution, its test statistic value will be used to determine if it can be generated by the null hypothesis.

According to Table 2.1, both density and likelihood ratio requires calculation of the cluster's area in Euclidean space. In the framework of spatial scan statistics, cluster regions are pre-defined (e.g., all circular regions of certain areas) so it is trivial to calculate the areas (e.g., πr^2). However, area calculation is not well-defined in the DBSCAN framework whose output clusters are represented by "maximal point sets". While conceptually (or visually in low-dimension space) it is easy to sense the region covered by a point-set cluster, mathematically it becomes tricky to model such an area. For example, a convex hull is a popular model to depict the region covered by a point set. However, since DBSCAN clusters can have arbitrary shapes (e.g., concave), a convex

hull modeling will introduce large errors into the estimation. An alternative is to use the ϵ neighborhoods of all core points in a cluster to approximate its area. However, this requires computation of the union of all these ϵ neighborhoods. Even for the two dimensional spatial case, just one area calculation will take $O(|c|^2 \log |c|)$ time where $|c|$ is the number of core points (or ϵ neighborhoods). It will require higher complexity and huge amount of time in higher-dimensional space.

Each test statistic involves normalization to make different clusters comparable [16]. For example, density ($d = n/a$) uses cluster area a as a normalization so it can be used to rank clusters with different areas a and number of points n, when areas are computable or available. One major disadvantage of density is its strong bias towards small clusters. Given a cluster of density d_0, there always exists a sub-region of it that has a higher density d_1. This means that a cluster of highest density will always have the smallest area, which is not a desired property [91, 16]. To reduce the effect, likelihood ratio incorporates the null hypothesis into the normalization and is able to reduce the bias. However, it ignores the phenomenon of spatial nondeterminism, making it still susceptible to the bias and leading to incorrect rankings of candidates [14, 16] (e.g., a well-known issue is that it includes tiny patterns in its results).

Cluster size n (i.e., number of points in the cluster) is another measure being used in scan statistics methods [16] that does not require computation of the area. To make clusters comparable, it does require some normalizing or constraining conditions to be enforced into the cluster search process; otherwise a bigger set of points is always superior. For DBSCAN, the normalizing conditions come naturally through the required parameters $(\epsilon, minPts)$. The search radius ϵ and minimum number of points $minPts$ clearly define the conditions that valid cluster points must satisfy, so the size of valid clusters (point sets) n are constrained by the conditions.

In the present study, our significance modeling uses cluster size n as the test statistic

because of the computational benefits (e.g., not requiring area computation).

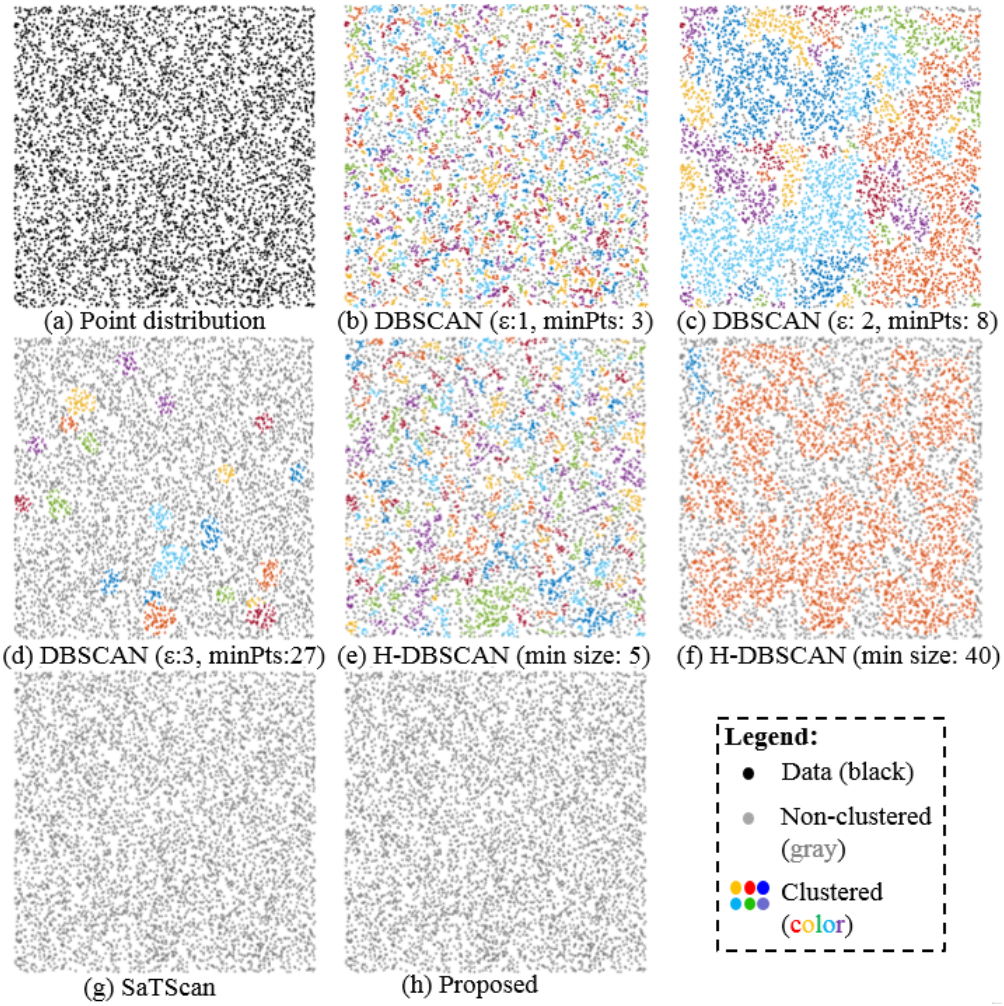


Figure 2.1: 2000-point data by a homogeneous point process.

In [88] (joint work by authors of DBSCAN and OPTICS), one issue discussed is the existence of "small" clusters in the results, which are likely to happen by chance. Here we extend this good start by formalizing the definition of "small" using statistical significance. Previously, to mitigate the "small" cluster issue, a remedy used by some users is to enforce a hard-threshold on minimum cluster size (e.g., default "5" in [90]).

While intuitively small clusters (e.g., with only 3 points) are likely to be chance patterns, chance patterns are not necessarily small. Fig. 2.1 shows the results of DBSCAN and HDBSCAN on a random point distribution generated by a homogeneous point process. In this point distribution, all clusters detected are chance patterns. Although the chance patterns are indeed small in a few results (e.g., Fig. 2.1 (b)(c)(e)), they turn out to be quite large (e.g., thousands of points) in others. Thus, the exact definition of "small" has to depend on many factors, such as the input data, the DBSCAN (or HDBSCAN) parameters, the desired significance level and the null hypothesis.

In this chapter we use significance testing to identify the exact threshold of "small" (i.e., minimum cluster size n_{min}) under all these factors to remove chance patterns (e.g., Fig. 2.1(h)).

2.2.3 Formal Problem Formulation

Inputs:

(1) A distribution of N geo-located points;

(2) DBSCAN parameters: (ϵ, $minPts$);

(3) A test statistic;

(4) A significance level α.

Output: Significant DBSCAN clusters (if they exist).

Objectives: Solution quality and computational efficiency.

Constraints: Correctness and completeness.

This formulation shows the main scope of the chapter, which is to enable significant DBSCAN clustering. While the formulation is defined using a single pair of $(\epsilon, minPts)$, later in Sec. 2.3.3 we will show how this formulation can be used as a sub-routine to detect clusters of varying densities in the context of significance testing (i.e., requiring enumeration of multiple ϵ and $minPts$). The test statistic we use is the cluster size n.

The completeness and correctness require that the clusters detected must satisfy the DBSCAN conditions and all cluster candidates that satisfy the conditions should be enumerated. This will be guaranteed by using exact DBSCAN as a sub-routine during significance testing. Note that while the formulation is for DBSCAN, it can be applied generally to model significance in its variations (e.g., HDBSCAN with the same test statistic), assuming that the outputs are also clustered point sets.

2.3 Significant DBSCAN Clustering

In this section, we first propose a baseline Monte Carlo method to evaluate the statistical significance of clusters detected by DBSCAN with a single pair of $(\epsilon, minPts)$ in Sec. 2.3.1. Then we propose a Dual-Convergence algorithm to accelerate the significance testing in Sec. 2.3.2. Finally, we present a heuristic search strategy which uses the single pair version as a sub-routine to detect significant clusters of varying densities in Sec. 2.3.3.

2.3.1 Significance Testing for A Single (ϵ, $minPts$)

Here we discuss significance testing for a given pair of $(\epsilon, minPts)$ in DBSCAN. Denote the significance level as α (e.g., 0.01, 0.05), the size of a detected cluster C as n_C, the total number of points in the point distribution as N, and the spatial domain of the point distribution as S. Denote $p_{null}(n_C, N, S, \epsilon, minPts)$ as the probability of having a cluster of size n_C or greater in a N-point distribution in S generated by a homogeneous point process. p_{null} is also the p-value.

Definition 1. *Cluster C is statistically significant if its p-value $p_{null}(n_C, N, S, \epsilon, minPts) < \alpha$.*

Currently there still does not exist a known statistical model that can calculate

the probability p_{null} in closed-form, because the calculation of the probability needs to consider the search and expansion process of DBSCAN as well as the randomness associated with distributing N points in a spatial domain S which can have irregular shape (e.g., a sub-space of a city in Sec. 2.4.1). Thus, we use a Monte Carlo method to estimate p_{null}.

A Baseline Algorithm with Monte Carlo Estimation

In Monte Carlo estimation (Alg. 1), we generate M simulation trials to approximate the distribution of cluster size n (i.e., the test statistic) in point distributions generated by a homogeneous point process.

In each trial, we first generate a random N point distribution using homogeneous point process, and then run DBSCAN with the same input $(\epsilon, minPts)$ to get the best or maximum cluster size $\hat{n}$ in the trial. After M trials, we will have M best $\hat{n}$ values from the trials. By sorting the M values in a descending order, we can estimate the p-value p_{null} of a cluster C detected from the real data by checking its rank r in the sorted list: $p_{null}(n_C, N, S, \epsilon, minPts) = r/M$. Note that M has to be at least $1/\alpha$ to evaluate the significance.

We reject the null hypothesis and mark cluster C as significant if $p_{null} < \alpha$ (or $r < \alpha M$). Equivalently, we just need to find the $(\alpha M)^{th}$ largest value in the sorted list and use that as a threshold (denoted as n_α) of cluster size to filter out non-significant clusters.

2.3.2 A Dual-Convergence Algorithm

The output of DBSCAN often contains multiple clusters with different sizes (i.e., number of points) and p-values. The baseline algorithm finds a threshold for cluster size n_α using the significance level α to classify the detections into significant $(n > n_\alpha)$ and

Algorithm 1 Monte Carlo estimation of n_α

Require:

- total number of points N and spatial domain S
- DBSCAN parameters $(\epsilon, minPts)$
- significance level α and number of Monte Carlo trials M

1: assert($M \geq 1/\alpha$)
2: $nList$ = new List(M)
3: **for** $i = 1$ to M **do**
4: $\quad data_r$ = getRandomPointDistribution(N, S)
5: $\quad clusters$ = DBSCAN($data_r$, ϵ, $minPts$)
6: $\quad nList(i)$ = max($clusters$.getSizes())
7: **end for**
8: $nlist$ = $nlist$.sort('DESC')
9: **return** $n_\alpha = nList(ceil(\alpha \cdot M))$

non-significant ($n \leq n_\alpha$) groups.

Finding the exact value of n_α in the baseline algorithm requires executing the exact DBSCAN algorithm across all M trials. Our idea is to reduce the number of exact DBSCAN runs in the Monte Carlo trials by generating bounds on the size n_{max} of the largest cluster (line 6, Alg. 1) in a simulated data (line 4, Alg. 1).

Since DBSCAN detections in real data may likely be a mixture of significant and non-significant clusters, an acceleration has to consider both cases to be truly effective. The proposed Dual-Convergence algorithm achieves this with: (1) an upper bound of n_{max} to accelerate the validation of significant clusters (Sec. 2.3.2); (2) a lower bound of n_{max} and an early-termination technique with a probabilistic performance guarantee to accelerate the filtering of non-significant clusters (Sec. 2.3.2 and 2.3.2); and (3) a dual-convergence framework which makes the above techniques work together to maximize

the speed-up (Sec. 2.3.2).

Upper and Lower Bounds of n_{max} with a Discrete Scan

To simplify the illustration, here we first consider the case of testing the significance of a single cluster with size n_C, which is detected from real-data. The general case will be detailed in Sec. 2.3.2.

In the baseline algorithm, in each round it finds the exact n_{max}. After M trials, if the total number of trials with $n_{max} \geq n_C$ is smaller than αM (equivalently $n_\alpha < n_C$), then n_C is significant; otherwise, non-significant.

Here our goal is to design an efficiently-computable upper bound $UB(n_{max})$ and lower bound $LB(n_{max})$ of n_{max} to avoid the need of exact DBSCAN if $n_C > UB(n_{max})$ or $n_C \leq LB(n_{max})$.

DBSCAN uses a circular ϵ-neighborhood to find core points and merge them into clusters through range queries. Denote ALG_{scan} as a more general version of DBSCAN, which may use other neighborhood definitions for finding core points and performing range queries for merging. Denote n'_{max} as the size of the largest cluster returned by ALG_{scan}. Lemmas 1 and 2 show the sufficient conditions (not necessary conditions) for an ALG_{scan} to make n'_{max} an upper or a lower bound of n_{max} from DBSCAN, respectively.

Lemma 1. *$n'_{max} \geq n_{max}$ if the neighborhoods defined in an ALG_{scan} (may vary from point to point) always fully **contain** the ϵ neighborhoods of DBSCAN as subspaces.*

Proof. The proof is straightforward. The core point set of ALG_{scan} must be a superset of that of DBSCAN, and any two core points merged by DBSCAN's range query must also be merged by ALG_{scan}'s because any ϵ-neighborhood of DBSCAN is always fully contained by the corresponding search neighborhood from ALG_{scan}. □

Lemma 2. *$n'_{max} \leq n_{max}$ if the neighborhoods defined in an ALG_{scan} (may vary from point to point) are always fully* ***contained by*** *the ϵ-neighborhoods of DBSCAN.*

Proof. The proof is symmetric to that of Lemma 1 by replacing supersets by subsets. Details skipped to avoid redundancy. □

We use a discrete-scan to build and compute the upper and lower bounds. In the discrete scan, we discretize the continuous spatial domain S into a grid where the value of each grid cell is the number of points inside (no need to store the actual points, just a numeric value). The length of a grid cell is a fraction of the neighborhood size ϵ from DBSCAN (e.g., 1/4).

Fig. 2.2(b) illustrates the definition of the neighborhoods in the discrete-scan to construct the upper bounds (Lemma 1). Denote G as a grid with with I rows and J columns generated from the discretization, $G(i, j)$ as a cell at the i^{th} row and the j^{th} column (row and column IDs start with 1), and $G(i_0 : i_1, j_0 : j_1)$ as a sub-grid containing all the cells with row $i \in [i_0, i_1]$ and column $j \in [j_0, j_1]$. Thm. 1 shows the neighborhood definition in discrete-scan that satisfies the sufficient condition for upper-bounding.

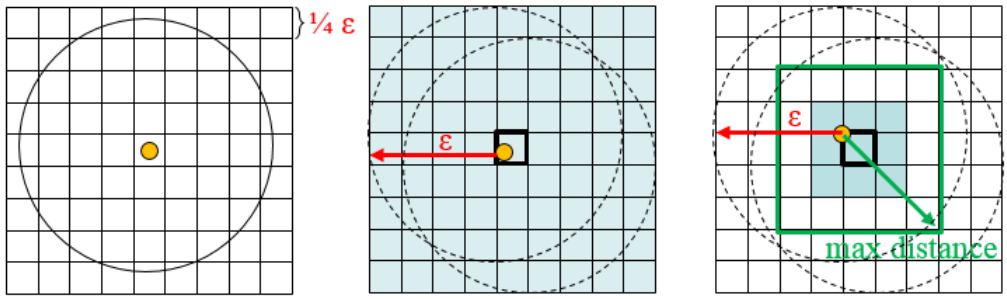


Figure 2.2: Sub-grids of upper and lower bounds.

Theorem 1. *For any point within $G(i, j)$, its circular ϵ neighborhood is always fully contained by the square neighborhood covered by $G(i - \Delta i : i + \Delta i', j - \Delta j : j + \Delta j')$,*

where $\Delta i = \min(\lceil \frac{\epsilon}{L} \rceil, i-1)$, $\Delta i' = \min(\lceil \frac{\epsilon}{L} \rceil, I-i)$, $\Delta j = \min(\lceil \frac{\epsilon}{L} \rceil, j-1)$, *and* $\Delta j' = \min(\lceil \frac{\epsilon}{L} \rceil, J-j)$.

Proof. Every point in the original space must be contained by a grid cell (e.g., the center cell of Fig. 2.2(b)(a)). Note that in order to make sure there is strictly no overlaps between cells, each cell only contains its top and left boundaries, except for those at the right-most columns or bottom rows. As shown in Fig. 2.2(b), a point in a cell lies at most on or infinitely next to the boundaries of a cell. Thus, its ϵ search distance in the continuous space is at most $\lceil \frac{\epsilon}{L} \rceil$ cells in the discrete space. As a result, the circular ϵ neighborhood must be fully contained by a sub-grid whose length is $(2 \cdot \lceil \frac{\epsilon}{L} \rceil + 1)$ cells centered at $G(i,j)$. The min function is used to constrain the sub-grid to be within G. □

Similarly, Thm. 2 shows the neighborhood definition that satisfies the sufficient condition for lower bounding.

Theorem 2. *For any point within* $G(i,j)$, *its circular* ϵ *neighborhood is always fully contained by the square neighborhood covered by* $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$, *where* $L \leq \frac{\epsilon}{\sqrt{2}}$, $\Delta i = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, i-1)$, $\Delta i' = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, I-i)$, $\Delta j = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, j-1)$, *and* $\Delta j' = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, J-j)$.

Proof. Here we need to guarantee that the sub-grid neighborhood is fully contained by the ϵ neighborhood of any point within $G(i,j)$. In other words, the furthest distance from a point in $G(i,j)$ to any location within the sub-grid neighborhood must be smaller than ϵ. As shown in Fig. 2.2(c), the maximum distance is achieved between a point in $G(i,j)$ located at its corner and another location at the sub-grid's furthest corner. This distance is at most $\frac{k-1}{2} \cdot \sqrt{2}L$, where k is an odd number representing the side length (unit: cell) of the sub-grid neighborhood. Since $(\frac{k-1}{2}+1) \cdot \sqrt{2}L \leq \epsilon$, we have k to be at

most $\left\lfloor \frac{\sqrt{2}\epsilon}{L} - 1 \right\rfloor$. Thus, the search distance outside $G(i,j)$ is at most $\frac{k-1}{2} = \left\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \right\rfloor$ cells. □

According to Theorems 1 and 2, we can construct the upper and lower bounds of n_{max} by using the grid-based neighborhoods. Note that the only difference in the neighborhood definitions for the upper and lower bounds lies in the values of Δi, $\Delta i'$, Δj and $\Delta j'$. Other than the side-lengths of the sub-grid based neighborhoods, the steps in the discrete-scan are exactly the **same** for the upper and lower bound calculation. Thus, in the following we will use $G(i - \Delta i : i + \Delta i', j - \Delta j : j + \Delta j')$ in a general manner (i.e., defined by either the upper or lower bound) to illustrate the key steps in the discrete-scan.

In the discrete-scan, we consider all the points within $G(i,j)$ as a single super-point whose (1) location is represented by the spatial extent of $G(i,j)$; (2) search neighborhood is covered by $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$; and (3) cardinality is equal to the number of original points within $G(i,j)$. The cardinality will be used only when determining the core points and computing the sizes of the clusters. Again, the core points here will be a superset of the core points in DBSCAN.

We use super-points and grid-based neighborhoods to perform the discrete-scan. Since super-points can be represented by grid cells $\{G(i,j)\}$, we only need to enumerate through grid cells instead of the actual points after the grid is constructed. Then, for each $G(i,j)$, the discrete scan uses the sub-grid $G(i - \Delta i : i + \Delta i', j - \Delta j : j + \Delta j')$ (Thm. 1 and 2) as the search neighborhood to determine if $G(i,j)$ is a core super-point:

$$G(i,j) \text{ is } \begin{cases} \text{core}, & \text{if } \sum_{G(i,j)\in g} |G(i,j)| \geq minPts \\ \text{not core}, & \text{otherwise} \end{cases} \tag{2.1}$$

where $g = G(i - \Delta i : i + \Delta i', j - \Delta j : j + \Delta j')$ and $|G(i,j)|$ is the cardinality of the super-point.

The merging or expansion process in the discrete scan also uses the sub-grid based neighborhoods as illustrated in Alg. 2. Note that Alg. 2 describes the expansion for a single super-point $G(i, j)$, and its output n_C is the total number of points in the cluster containing $G(i, j)$. In line 11 of Alg. 2, the Hadamard Product ($cluster \circ G$) is the matrix multiplication performed in element-wise fashion. As a reminder, the value of a cell in G is the number of points in it.

Algorithm 2 Expansion of a single $G(i, j)$ in a discrete-scan

Require:

- Grid G and a core super-point $G(i, j)$
- Helper binary grid (visited: 1, non-visited: 0) V
- DBSCAN's minimum number of points $minPts$

$queue$ = Queue().enqueue($G(i, j)$)

{# Mark the cells belonging to this cluster}

$cluster$ = new Grid(size=G.size; value=0)

while !empty($queue$) **do**

  $core$ = $queue$.dequeue()

  g = getGridNeighborhood(G, $core$)

  $cores_{new}$ = findNewCoresInNeighbor($core$, g, $minPts$, V)

  $queue$.enqueue($cores_{new}$)

  updateVisitedCells(V, g)

  markClusterCells($cluster$, g, newValue=1)

end while{# Get cluster size.}

n_C = computeHadamardProduct($cluster$, G).sum()

return n_C, V

After calculating the cluster sizes for all clusters, we can easily compute the bound on n_{max}. Note that we need to run the discrete-scan twice with different neighborhood

definitions (i.e., Thm. 1 and 2) to compute both $UB(n_{max})$ and $LB(n_{max})$. Given the size of a cluster n_C from real data, we can avoid running the exact DBSCAN if $n_C > UB(n_{max})$ or $n_C \leq LB(n_{max})$.

Early Termination with Theoretical Guarantee

Early termination has been widely used to stop Monte Carlo estimation with $M' < M$ trials. For example, if there are already (αM) trials with $n_{max} \geq n_C$, then the cluster of size n_C must not be significant and we can directly terminate it without performing the rest trials.

In related work, early termination is mostly used as a simple heuristic as its effectiveness is not well understood in theory.

In this work, early termination will be a key building block in the dual-convergence process (Sec. 2.3.2) to filter out non-significant clusters. Thus, we provide a detailed theoretical analysis of early termination and shows the probabilistic performance estimation of this technique for non-significant clusters.

Denote M as the number of Monte Carlo trials and α as the significance level ($M \geq 1/\alpha$). Denote n as the cluster size of a non-significant cluster and $p(n)$ as the probability of having $n_{max} \geq n$ in a random point distribution generated by a homogeneous point process. We first develop the following Lemma 3.

Lemma 3. *For cluster size n with $p(n)$, the probability of terminating within the x trials is:*

$$1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot [p(n)^i \cdot (1 - p(n)^{x-i})] \tag{2.2}$$

Proof. First, having exactly i trials with $n_{max} \geq n$ in x trials follows a binomial distribution, so we have its probability as $f_{bin}(i, x, p(n)) = \binom{x}{i} \cdot [p(n)^i \cdot (1 - p(n)^{x-i})]$. Then, the probability of terminating within x trials is equivalent to having at least αM

trials with $n_{max} \geq n$ in x trials. Thus, this probability follows the cumulative binomial distribution. □

Lemma 3 is applied for a specific non-significant cluster size n and requires knowing $p(n)$. To make it useful in practice, we remove the $p(n)$ requirement by integrating Eq. (2.2) across all possible values of $p(n)$ and calculating the expectation:

Theorem 3. *For a non-significant cluster, the probability of terminating within the x trials is lower bounded by:*

$$P_{early}(\alpha M, x) = 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot \frac{i!(x-i)!}{(x+1)!} \tag{2.3}$$

Proof. By integrating Eq. (2.2) across all $p(n)$ we have:

$$\begin{aligned} P_{early}(\alpha M, x) &= \int_0^1 \left\{ 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot [p^i \cdot (1-p)^{x-i})] \right\} dp \\ &= 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot Beta(i+1, x-i+1) \\ &= 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot \frac{i!(x-i)!}{(x+1)!} \end{aligned}$$

where $Beta(i+1, x-i+1)$ is the Beta function with two inputs. □

Suppose we have $\alpha = 0.01$ and $M = 100$. The probability of terminating after the 2^{nd} trial is 0.67, 3^{rd} is 0.75 and 10^{th} is 0.91. Similarly, for $\alpha = 0.01$ and $M = 1,000$, then the probability of terminating after the 20^{th} trial is 0.52, 30^{th} is 0.68 and 100^{th} is 0.90. This shows that early-termination can be very effective.

Dual Convergence towards the Significance Boundary

To summarize, the purpose of the upper bound is to accelerate the validation of significant clusters and the purpose of the lower bound and early-termination is to speed-up the filtering of non-significant ones.

Our previous discussion mainly concerned acceleration of significance testing for a single cluster size. In a real-world scenario, DBSCAN often returns a list of clusters with different sizes. Denote D as a list of the sizes of detected clusters from a real dataset. D is sorted in a descending order. There are three possible scenarios of cluster composition in D: (1) D contains only significant clusters; (2) D contains only non-significant clusters; and (3) D contains a mixture of significant and non-significant clusters. The first two scenarios are relatively easier because they can achieve acceleration from a single technique (Sections 2.3.2 and 2.3.2).

For the third scenario, the techniques need to share information and work together to really reduce computational cost. For example, the upper bound avoids having to run the exact DBSCAN if $n_C > UB(n_{max})$, where n_C is the size of a detected cluster from real data. Since we have multiple sizes in D, the condition becomes $\min(D) > UB(n_{max})$ in order to avoid the exact DBSCAN. When D contains non-significant clusters, this condition may be rarely satisfied, making the upper bound ineffective. Similarly, early termination is not expected to be effective when there is at least one significant cluster in D because this means we have to perform at least $(1-\alpha)M$ trials (e.g., 99% of the trials for $\alpha = 0.01$).

To address this issue, we present a dual-convergence framework to coordinate the techniques and make them increasingly efficient as the trials progress.

Fig. 2.3 shows the progression of the trials and the information sharing among the techniques. Here early-termination controls a pointer et^* on D to mark the current boundary of non-significant clusters. Accordingly, when the upper bound operates in

a trial, it can skip the exact DBSCAN if the size at et^* is greater than $UB(n_{max})$, i.e., $D(et^*) > n_{max}$ instead of $min(D) > n_{max}$. As more trials complete (e.g., 5% of M according to the analysis in Sec. 2.3.2), et^* will gradually cover all non-significant clusters and the upper bound will approach its maximal effectiveness. Note that the upper-bounded sizes are trial-specific.

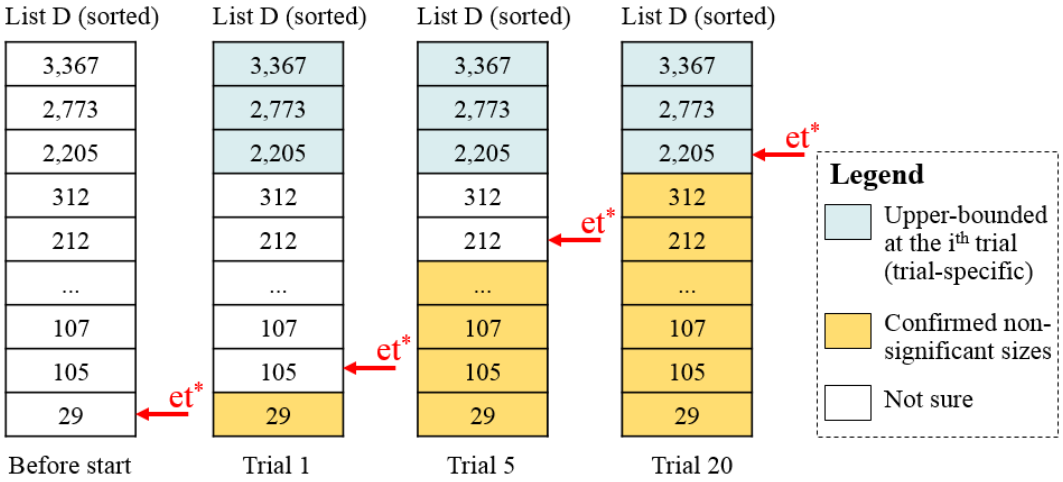


Figure 2.3: Progression along trials in dual convergence.

Algorithm 3 shows the execution order of the three techniques, i.e., upper bound, lower bound and early termination.

Time complexity

Due to the space limit, we show the time complexity of the significant DBSCAN without detailed proofs. Denote N as total number of points, M as number of trials, and $|G|$ as number of grid cells in the discrete scan. The time complexity of the baseline algorithm is $O(f_{DB}(N) \times M) = O(MN \log N)$ in a 2D spatial space, where $f_{DB}(N)$ is the complexity of DBSCAN that can differ by data dimensions. This chapter focuses on the spatial case. The time complexity of the Dual-Convergence algorithm is $O(\rho MN \log N + (1-\rho)M \cdot$

$\max(|G|, N))$, where $\rho \in (0, 1]$ is the portion of trials that requires the exact DBSCAN. Note that in a 2D spatial case, as N increases we normally have $|G| << N$. The discrete scan still requires at least linear time because we need to count the number of points in each grid cell. This needs a single and simple pass through the actual points (the cell a point belongs to is trivially computable in closed-form). After that, it is just a quick linear scan through the cells (i.e., $O(|G|)$ time), which no longer involves any computation with the actual points or other special data structures (can be used with integral image for acceleration). In experiments (Sec. 2.4.2) we will see that the discrete scan works very well (overhead is almost negligible). For non-clustered data, we also expect the early termination to be very effective according to the probabilistic analysis (Sec. 2.3.2) and experiment results (Sec. 2.4.2). In a high-dimensional space, the overall computation is still challenging because grid-based methods normally have time complexity exponential to the variable dimension d (unless the complexity treats dimension d as a constant no matter how large it is, e.g., $d = 100$). Our current scope is the 2D spatial case.

2.3.3 Significant Clusters with Varying Densities

In this section we discuss a heuristic search strategy which uses significant DBSCAN for a single $(\epsilon, minPts)$ as a subroutine to detect clusters of different densities in the context of significance testing. The **main focus** here is to illustrate the benefits of significance testing and the way to perform it when considering various densities. The scope of this discussion is **not** to further mature related work in terms of detecting clusters of varying densities. Nonetheless, the comparisons to related work are shown through experiments.

In DBSCAN, in order to detect clusters of varying densities we need to consider a

range of *eps* and $minPts$ rather than a fixed pair. A key issue that rises when we aggregate DBSCAN results with multiple pairs of $(\epsilon, minPts)$ is that it may greatly increase the number of spurious detections (i.e., chance patterns from different $(\epsilon, minPts)$). This is where significance testing becomes more important by playing a key role in eliminating the spurious patterns.

In the following, we describe significance testing in this scenario with an example heuristic search strategy for multi-densities.

An Example Heuristic Search Strategy

The search strategy basically enumerates through a list of $(\epsilon, minPts)$ for various densities (i.e., $minPts/(\pi\epsilon^2)$). Since density calculation requires both the number of points and area, considering only a single scale (i.e., one ϵ) for each density may not be appropriate. To mitigate this, the search strategy additionally considers a range of ϵ for each density.

Denote U as a matrix where each row corresponds to a single density and each column represents a specific scale (i.e., ϵ). The value of each entry $U(i, j)$ is the $minPts$ value (rounded) of $\epsilon(j)$ for $density(i)$. Lower row IDs in U represent smaller densities and lower column IDs represent smaller scales. Starting at the highest density, the heuristic search follows two key steps in each search-round. **Step-1:** It fixes the density at the current row i and sequentially executes original DBSCAN on the same data but through different scales from low to high. It stops at the scale $U(i, j)$ when the increase in average cluster size from $U(i, j)$ to $U(i, j + 1)$ is smaller than $\lambda\%$ (e.g., 10%); **Step-2:** It fixes the scale at the j^{th} column (from step-1) and sequentially executes original DBSCAN towards lower densities. It stops at $U(i', j)$ when the average cluster size from $U(i', j)$ to $U(i' + 1, j)$ is smaller than $\lambda\%$ (e.g., 10%). The DBSCAN result at $U(i', j)$ is returned for this search-round (corresponding to a density-level). This two-step process

is used to avoid inappropriate densities (e.g., too high) or scales (e.g., too small) which may cause a single cluster to be shattered into pieces.

Using this example two-step (per round) search, we show how significance testing should be performed in the next section.

Significance testing

Significance testing is performed after each round on the returned result. To be consistent, the significance testing uses the same $(\epsilon, minPts)$ parameter pair used for the returned result. Note that here we only have a single pair of parameters and can directly perform the significance testing described in Sec. 2.3.2. In addition, the search strategy is only needed for real data and not the significance testing (single $(\epsilon, minPts)$ based).

After finding significant clusters, a **critical step** is to remove them from the data before the next round of search. Since the next thing we need to test is whether the rest of the data is generated by a homogeneous point process or it contains other significant clusters (e.g., those with lower densities), the analysis should be independent from any already-confirmed significant clusters. Thus, we need to remove both the points of the significant clusters and their spatial coverage. This will result in a smaller amount of points and a smaller spatial domain for future significance testing. The challenge in this step is the removal of the spatial coverage of significant clusters. Since the exact coverage of the clusters (point sets) is not defined in DBSCAN, we rasterize the continuous spatial domain into pixels and approximate the coverage of the clusters by marking the pixels within the ϵ neighborhoods of the core points. In subsequent significance testing, the points will only be randomly distributed to the unmarked sub-spaces.

After the removal, the heuristic search continues with the next (lower) density.

2.4 Validation

Fig. 2.4 shows the overall validation framework and questions.

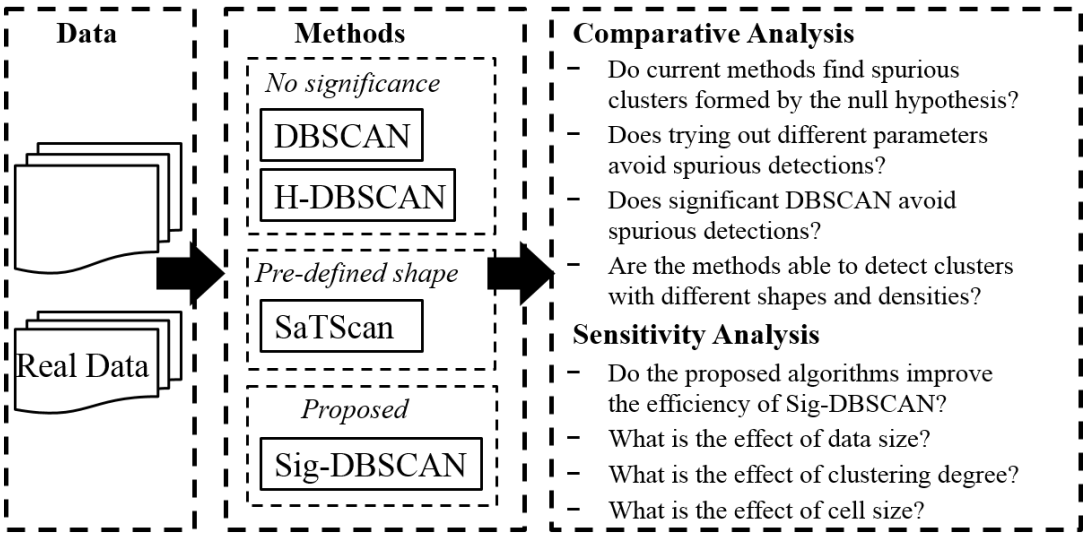


Figure 2.4: Overall validation framework.

2.4.1 Comparative Analysis

The candidate methods include DBSCAN with varying parameters, H-DBSCAN with varying parameters, SaTScan, and the proposed significant DBSCAN with significance level set to 0.01. Note that the significant DBSCAN used the example heuristic search strategy described in Sec. 2.3.3, and any parameters associated with it remain the same throughout the experiments for data with different number of points, cluster shapes, cluster densities, point processes, etc.

Performance on Data Generated by a Homogeneous Point Process

Fig. 2.1 (2,000 points, in Sec. 2.2.2) shows the results of the candidates methods on two datasets generated by a homogeneous point process (i.e., no true clusters). The

results of DBSCAN were evaluated on three sets of parameters $(\epsilon, minPts)$. Note that we could have possibly used parameters corresponding to very high densities and made sure there would be no clusters detected. However, this choice is not appropriate for two main reasons: (1) we cannot be sure whether or not a given data is truly clustered (i.e., it could possibly contain real clusters) without testing so it is not proper to assume it beforehand; and (2) the same parameters were tried on clustered data with 2,000 and 10,000 points in the same spatial domain in Sec. 2.4.1. We can see that some of these parameters already represented a density that is too high to correctly detect true clusters (e.g., Fig. 2.5(b)(d) and Fig. 2.6(b)(c)). Thus, the choice of parameters already covered sufficiently high densities.

H-DBSCAN [88] is considered as an improved cluster detection method that combines the strengths of DBSCAN and OPTICS[87] (made by the authors of the papers). By default, H-DBSCAN does not require input parameters and can automatically find the best clusters based on its criteria. If needed, a minimum cluster size can be set to refine the criteria. A common value for the minimum cluster size is 5 (e.g., defaulted in its standard Python library [90]). In our experiments, we tried both 5 and 40.

The results of both DBSCAN and H-DBSCAN contain spurious patterns for different parameters. For DBSCAN, in general the spurious patterns were smaller when the $(\epsilon, minPts)$ corresponds to a higher density (also affected by the scale). Some detections are quite large in cluster size so it is difficult to remove them with a pre-set threshold of minimum cluster size. We can see a similar trend for H-DBSCAN results. When a larger minimum cluster size is used, the detected spurious clusters also becomes larger, making them difficult to remove without significance testing. This also mirrors our earlier analysis in Sec. 2.2.2. In Fig. 2.1(g)(h) we can see that both SaTScan and the proposed significant DBSCAN can successfully avoid chance patterns with significance testing.

Performance on Data with Clusters of Varying Shapes

Fig. 2.5 (10,000 points) and Fig. 2.6 (2,000 points) show the results on two datasets generated by biased/clustered point processes. Each has four clusters of different shapes. As we can see, SaTScan is overall robust against the noises with significance testing, but it is limited in detecting clusters of general shapes. As a spatial scan statistic approach, SaTScan operates in a top-down fashion. In other words, it enumerates candidate regions of different sizes with a pre-defined geometric shape and checks their statistical significance. In addition, the enumeration space for a single shape is often already quite-large (e.g., circles of different sizes across all locations) and results in large computational cost. In contrast, DBSCAN operates in a bottom-up fashion, which allows it to trace arbitrary shapes automatically without predefined shapes. Note that it does assume the density within a cluster is generally homogeneous without drastic changes.

As we can see, in general DBSCAN based methods show stronger capacity than SaTScan in finding clusters with varying shapes, although some of their results still contain many spurious patterns due to the lack of significance testing. In addition, H-DBSCAN methods also seem to be sensitive to the minimum cluster size threshold. For example, in Fig. 2.6(f), the first three clusters were detected as a single one. In Fig. 2.5(h) and Fig. 2.6(h), the proposed significant DBSCAN is able to detect the significant clusters of arbitrary shapes while successfully avoiding chance patterns.

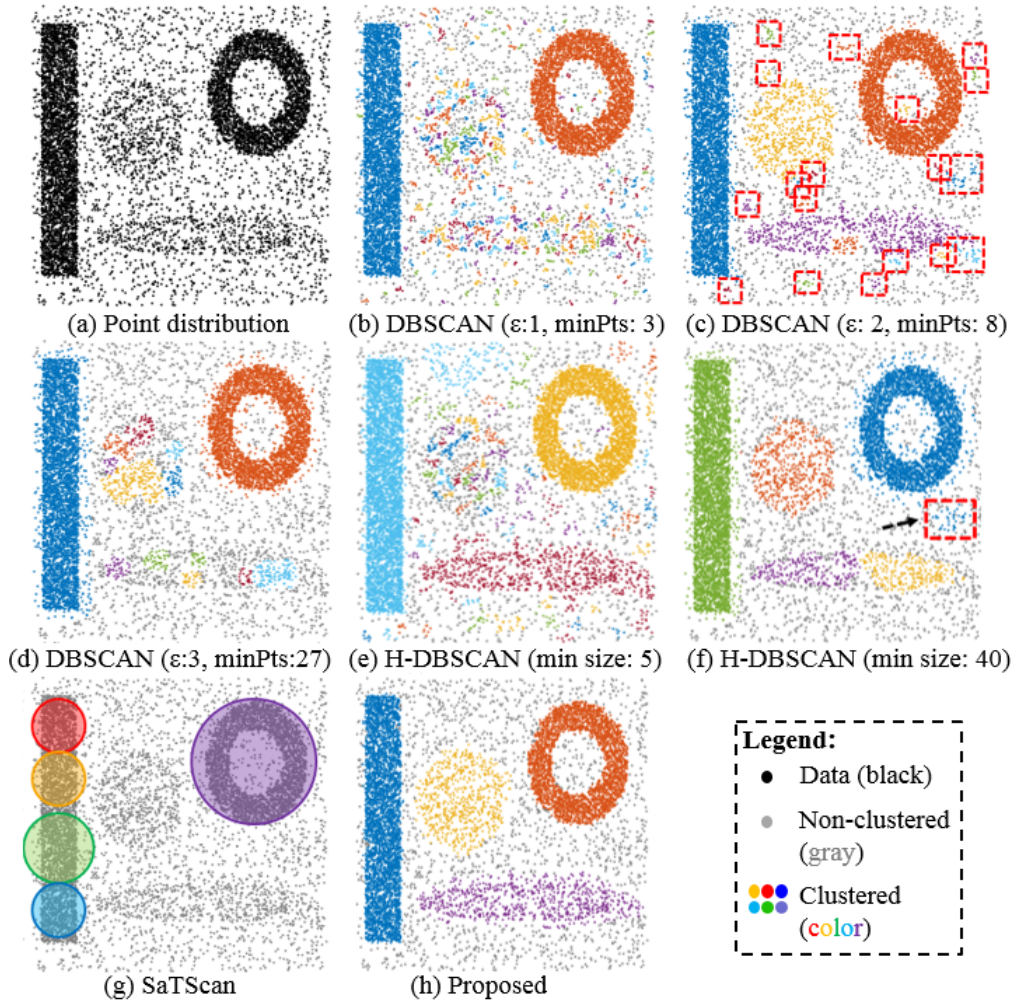


Figure 2.5: Clustered 10000-point shape data.

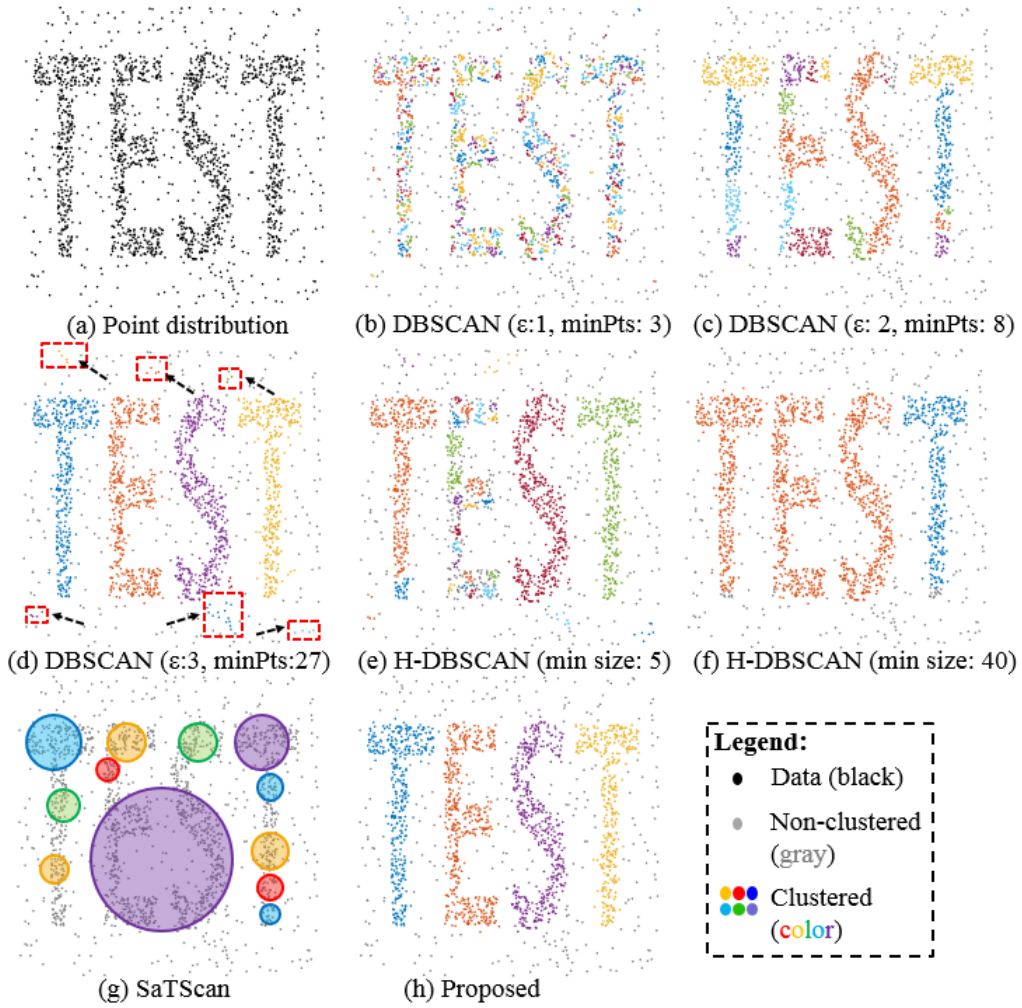


Figure 2.6: Clustered 2000-point test data.

Performance on Data with Clusters of Varying Densities

The four clusters in Fig. 2.5 have different densities. The probability density of having a point in the rectangle and ring is twice as high as that in the circle and ellipse.

As we can see, DBSCAN methods can detect true clusters of different densities by varying its parameters. However, the number of chances patterns also greatly increases if we merge their results. The same trend can be seen in the results of H-DBSCAN.

With significance testing, the proposed sig-DBSCAN method was able to filter out the chance patterns and keep only the significant clusters of varying densities (i.e., with the heuristic search in Sec. 2.3.3). Again, any parameter associated with the heuristic search remained the same with no change throughout the experiments on different data.

Real-world Example: Minneapolis Snow Emergency Tows

Fig. 2.7 shows the results of the candidate methods on an example real-world data of snow-emergence vehicle tows (948 points) in Minneapolis, US, 2019. The city is located at the north side of the US near the Lake Superior, and receives heavy snows in the winter. To plow the thick snow from the streets (i.e., curb to curb), a snow emergency requires all cars parked in the declared zones to be moved to other places; otherwise they will be tagged or towed.

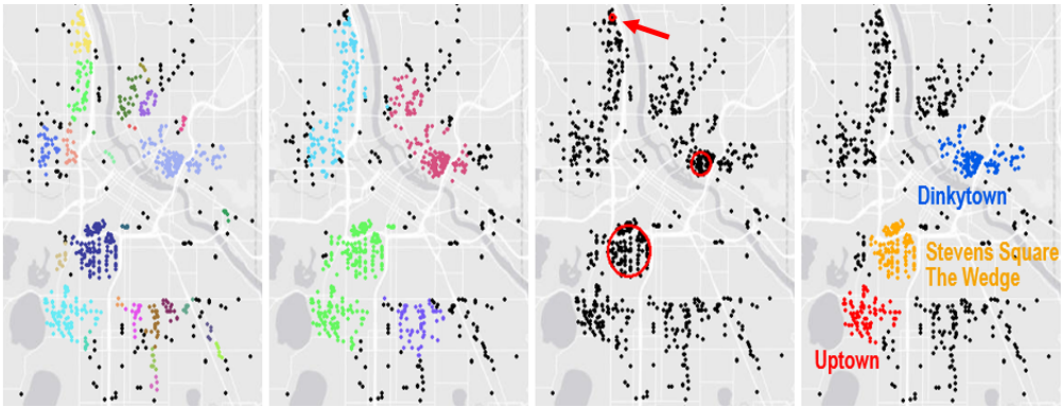


Figure 2.7: A real-world example: snow emergency tows.

Given that there are limited tow-trucks and time for towing, towing priority (e.g., more tow trucks, higher frequency) is given to neighborhoods that have narrower streets, which will be very difficult to use without complete snow shoveling. According to a transportation officer in Minneapolis (2019)[92], the neighborhoods with the most

pressing needs in the snow emergency were: Stevens Square, the Wedge, Dinkytown and Uptown. We used this real-world data and information to test if the methods can identify these priority neighborhoods in the snow emergency.

Note that the invalid regions (e.g., rivers, lakes, parks, urban forests, highway etc.) in the figure were excluded from the input spatial domain before running these methods. The study area used for the snow emergency data was approximated by the general coverage of the points. In the experiment results, we can see that the four neighborhoods are reasonably well captured by the significant DBSCAN (Fig. 2.7(d)). The clusters that contain the neighborhoods (sometimes together with a few adjacent ones) are highlighted with the corresponding names. In the DBSCAN result (Fig. 2.7(b)), we can still see these significant clusters, but there are many non-significant ones covering most of the points. In H-DBSCAN, the clusters are more contiguous with some being merged together but non-significant ones still persist in most of the study area. SaTScan (Fig. 2.7(c)) was able to find roughly two of the neighborhoods. It missed the one at Uptown, potentially due to the shape of the clusters (i.e., a large empty space towards the right-side when a circle is used to cover it). In the other two large clusters it identified, there is not much empty space left in the circles. In addition, SaTScan detected a very tiny cluster near the top-left, which is also a well-known issue (i.e., tiny clusters tend to have a very high likelihood ratio).

2.4.2 Sensitivity Analysis

We evaluated the computational performance of the baseline and the Dual-Convergence algorithms on various data sizes N, clustering degrees (or effect sizes) es and cell-sizes cs in discrete scan. For clustered data, the clustered regions are the same as those in Fig. 2.5. To add some density variations in the clustered data, the probability density of having points (modeled by es) in the rectangle and ring was set twice as high as

that in the circle and ellipse. The default parameter values used were $N = 10,000$, $es = (10, 20)$ and $cs = \epsilon/4$.

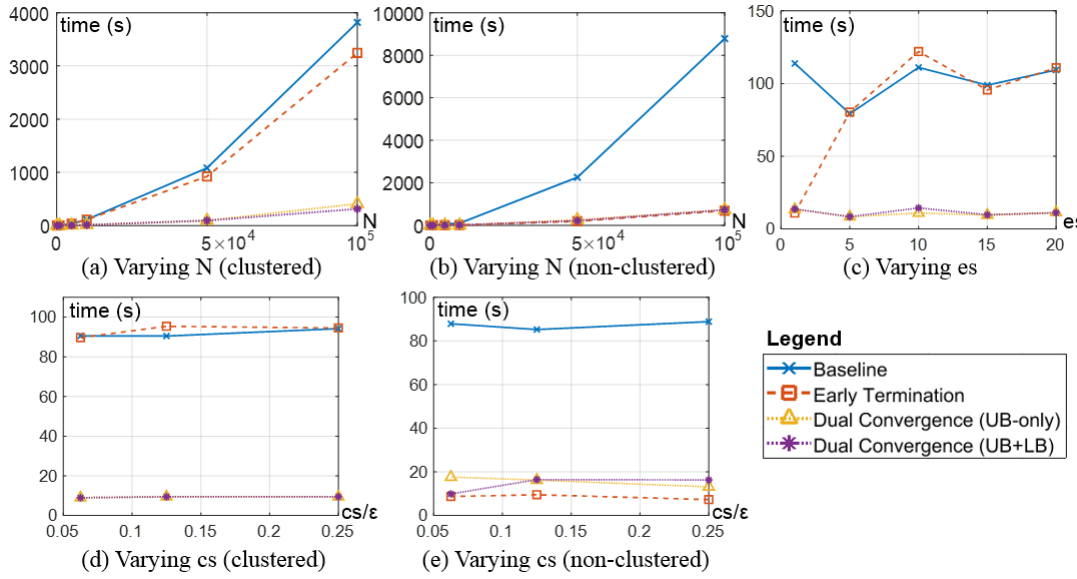


Figure 2.8: Execution time analysis.

Effect of Data Size

We evaluated the execution on varying data sizes N on both clustered (Fig. 2.8(a)) and non-clustered data (Fig. 2.8(b)). For both data types, the general trend is that the proposed Dual-Convergence method can greatly reduce the computational cost and the savings becomes greater as N increases. For early-termination, it is not very effective by itself when data is clustered because at least $(1-\alpha)M$ trials are needed. Comparing the upper-bound and lower-bound methods, we can see that the execution time savings are mostly achieved by the upper-bound alone while not much more is contributed by the lower bound. The reason might be that the lower bound is not as tight as the upper bound with the cell size used (Fig. 2.2 (b) and (c)).

Effect of Clustering Degree

Clustering degree (or effective size) es shows how much more likely an individual location within a clustered region will have a point compared to a location outside. Note that if the spatial coverage of a cluster is too small, even a large effect size can be hard to observe and confirm due to the small cumulative probability. We used two different es for each clustered data: two clusters will have es_0 and the other two will have $es_1 = 2 \cdot es_0$. The X-axis in Fig. 2.8(c) shows es_0. Note that for non-clustered data all the locations have the same $es = 1$.

The trend is that the Dual-Convergence algorithm consistently achieved great time reduction compared to the baseline throughout the experiment. The sharp increase of early-termination shows the switch from non-clustered $es = 1$ to clustered data $es > 1$.

Effect of Cell Size in Discrete Scan

Fig. 2.8(d) and (e) show the execution time under varying cell sizes cs in the discrete scan. Note that cs we used is a fraction of ϵ. Our expectation is that a finer cs can tighten the upper and lower bounds at the cost of increased number of cells in the discretization. We can see that overall the cost associated with the increase of number of cells is secondary (i.e., very small overhead) in terms of computation. In addition, there was a small improvement of lower bound from $cs/\epsilon = 1/8$ to $1/16$ while the improvement in the upper bound is not obvious. The reason could be that the upper bound is much tighter than the lower bound so it is less affected within this range of cs variation, whereas the lower bound can benefit a lot more with a smaller cs.

2.5 Conclusions and Future Work

We introduced, discussed and proposed a framework for incorporating statistical significance in DBSCAN clustering. To perform the significance testing, we proposed a baseline Monte-Carlo method as well as a Dual-Convergence algorithm for acceleration. In addition, we discussed cluster detection of varying densities (i.e., multiple $(\epsilon, minPts)$) in the context of significance testing. Our experiment results show that the proposed significant DBSCAN can greatly improve solution quality by robustly eliminating chance patterns. The Dual-Convergence algorithm also can greatly improve the computational efficiency.

Future work: This work is just a start of improving the robustness of widely-used clustering methods by modeling statistical significance. We expect the results to encourage more studies to explore significance testing of other methods (e.g., k-means). Also, many other opportunities exist to further improve this work, including different modeling strategies (e.g., advanced test statistics, hypotheses, underlying population), different DBSCAN methods (e.g., H-DBSCAN), higher dimensions beyond spatial, better strategies for multi-density search, analysis of significance and clustering quality (e.g., scale invariance, consistency, richness) [93], computational enhancements (e.g., distributed computing), etc.

Algorithm 3 Dual convergence

Require:

- List of cluster sizes D
- Number of Monte Carlo trials M
- Significant level α {#Other detailed inputs are **skiped**}

$et = D$.length

D_{geq} = new List(length=D.length) {# Tracking the number of trials with $n_{max} \geq D(i)$}

for $t = 1$ to M **do**

  UB = getUB() {#upper bound}

  if $D(et) \leq UB$ **then**

    $unbounded\_id$ = getMinZeroID($D > UB$)

    LB = getLB() {#lower bound}

    if $D(unbounded\_id) \geq LB$ **then**

      n_{max} = exact_DBSCAN()

      ids = getAllNonzeroID($D \leq n_{max}$)

      $D_{geq}(ids)$ += 1

    else

      $D_{geq}(unbounded\_id : end)$ +=1

    end if

    et = getMaxNonzeroID($D_{geq} < \alpha M$){#early term.}

    if et==None, **then** break, **end if**

  end if

end for

return $D_{sig} = (D_{geq} < \alpha M)$

Chapter 3

FF-SA: Fragmentation-Free Spatial Allocation

3.1 Introduction

Given a $m \times n$ grid G containing a set of orthogonal grid cells g_{ij} and a list L of choices, where each choice on g_{ij} has a benefit value b_{ij} and cost c_{ij}, the fragmentation-free spatial allocation (FF-SA) aims to find a tile-partition of G and a choice for each tile to maximize the overall benefit under a cost constraint and spatial constraints (e.g., minimum tile area, shape) which enforce spatial contiguity and regularity. FF-SA is related to both optimization and computational geometry.

FF-SA is an important problem in many application domains. In agricultural landscape design (a.k.a., Geodesign) [64, 94], land covers and management practices need to be determined for each spatial location in order to improve environmental or economic objectives. For example, maximizing food production under water quality constraint. The allocated land cover patches (e.g., continuous area of corn or wheat) must be geometrically large enough and regular in shape to allow efficient use of modern

farm equipment (e.g., commonly used 40-60 feet wide combine harvester heads in mid-western US). Without the spatial constraints, a spatial allocation result often contains land fragmentation (e.g., tiny land cover patches), which can significantly reduce farming efficiency [95, 96]. Similarly, in urban land-use planning and building floor zoning [97, 98], zones need also satisfy the spatial constraints to avoid undesired fragmentation, such as tiny nested zones of different types of land-uses (e.g., residential, industrial). FF-SA addresses these fragmentation issues using spatial geometric constraints.

FF-SA is approximation-hard (APX-hard), which implies NP-hardness. The APX-hardness shows that FF-SA has no polynomial time approximation scheme (PTAS) unless P=NP. The problem is computationally challenging given the large number of decision variables and constraints needed for real-world applications (e.g., million, trillion). Due to the hardness and scale of the problem, standard MIP solvers cannot efficiently solve FF-SA. Without modeling spatial constraints, conventional integer programming formulations have also been used for spatial allocation problems [94, 97, 98]. However, grid cells are treated as independent variables during optimization and this leads to fragmentation in the final results. In Geodesign Optimization [10], a dynamic growth tiling framework (DGTF) was used to avoid fragmentation during tiling scheme generation. The limitation of DGTF is that it only enumerates one full tiling scheme in the heuristic optimization process, which means its enumeration space of tiling scheme is very small. The new approach targets improving the solution quality of FF-SA by significantly increase the enumeration space of tiling schemes during optimization. Details are discussed in Sec. 3.4 and 3.5.

In this work, we prove that the hardness class of FF-SA is at least APX-hard beyond NP-hard, which further encourages research on heuristic solutions instead of exact or approximation algorithms. Then, we propose a new algorithm based on dynamic programming, namely Hierarchical Fragmentation Eliminator (HFE), to better approach

an upper bound on optimal solution quality by significantly increasing the enumeration space of tiling schemes. An acceleration algorithm based on multi-layer integral image is also proposed to improve the performance of the new approach. The computational time complexity of the new algorithms are analyzed in details.

A detailed case study in agricultural spatial allocation was performed to evaluate the proposed algorithms. The case study was carried out at Seven Mile Creek watershed (Minnesota, US) which has an area of 25,000 acres. Experiment results show that: (1) the HFE algorithm was able to eliminate fragmentation existed in conventional integer programming (no spatial constraints) solution; (2) HFE consistently produced better spatial allocation solutions compared to DGTF [10] in the study area; (3) the acceleration algorithms significantly improved the computational performance of HFE.

3.2 Problem Statement: Fragmentation-Free Spatial Allocation

The problem formulation for FF-SA inherits the formulation of Geodesign Optimization problem [10] and is generalized to a broader domain. Some key concepts are defined as follows: **(1) Grid:** A grid partition contains m rows and n columns. Each grid cell is identified by its row i and column j. Each grid cell g_{ij} has d choices, and each choice k has a benefit b_{ijk} and cost c_{ijk}. The choices for each grid cell are the same, but the benefit and cost values for each choice vary across grid cells. **(2) Tile:** A tile is rectangular in shape. Each tile $tile_t = (i_0, j_0, i_1, j_1)$ is identified by its top-left grid cell (i_0, j_0) and bottom-right grid cell (i_1, j_1). The choice made on all grid cells within a tile is the same. For choice k on a tile $tile_t$, the cost c_{tk} and benefit b_{tk} are the sum of b_{ijk} and c_{ijk} of the grid cells g_{ij} inside $tile_t$. Given N grid cells, N^2 tiles can be uniquely defined. **(3) Tiling scheme:** A tiling scheme is a tile-partition of the study area. Tiles

in a tiling scheme must not have overlaps. FF-SA is formulated as:

Inputs:

- A grid G and a list L of choices;
- A list Z contains all combinations of tiles and choices with the corresponding benefit and cost: $\{tile_t : (i_0, j_0, i_1, j_1),\ choice : k \in L, b_{tk}, c_{tk}\}$;
- A limit on total cost c_{tot}, minimum tile area α, minimum tile width (also height) β.

Output: A tiling scheme with choice assignment, which is a subset Z' of Z.

Objective: Maximize the total benefit: $\sum_{p=1}^{|Z|}(Z_p.b) \cdot s_p$, where $Z_p.b$ is the benefit value of Z_p, and s_p is 1 if $Z_p \in Z'$ and otherwise 0.

Constraints:

- Binary value constraint on s_p: $s_p \in \{0, 1\}$
- Each tile can only have one choice: $\sum_{p \in V} s_p = 1, V = \{p | Z_p.choice = k\}, \forall k$
- Total cost is less than or equal to c_{tot}: $\sum_{p=1}^{|Z|}(Z_p.c) \cdot s_p \leq c_{tot}$
- Each element Z'_p in Z' must satisfy a minimum area α and width β:

$$i_1 - i_0 + 1 \geqslant \beta, i_1, i_0 \in Z'_p \tag{3.1}$$

$$j_1 - j_0 + 1 \geqslant \beta, j_1, j_0 \in Z'_p \tag{3.2}$$

$$(i_1 - i_0 + 1) \cdot (j_1 - j_0 + 1) \geqslant \alpha, i_1, i_0, j_0, j_1 \in Z'_p \tag{3.3}$$

- There is no spatial overlap among elements in Z':

$$\forall Z'_1, Z'_2 \in Z', Z'_1.tile \cap Z_2.tile = \phi \tag{3.4}$$

Constraints (1) to (4) are spatial constraints imposed on tiles to avoid fragmentation. Constraints (1) to (3) are single-tile constraints. For implementation, these constraints

can be imposed by preprocessing, which removes Z_p in Z if $Z_p.tile$ does not satisfy the area and width constraints. Constraints (4) is a pair-wise constraint. A concrete mathematical formulation of (4) requires adding constraints (5) and (6) for all pairs of elements in Z (either (5) **or** (6) needs to be satisfied). Two elements Z_x and Z_y in Z are spatially disjoint only if none of the vertices in Z_x's tile falls into Z_y's tile. This indicates that each vertex (i, j) of Z_x's tile must satisfy the following non-linear constraints when $Z_x \in Z'$ and $Z_y \in Z'$:

$$(i - i_0) \cdot (i - i_1) > 0,\ i_1, i_0 \in Z_y \ or\ s_x = 0 \ or\ s_y = 0 \tag{3.5}$$

$$\textbf{or}\ (j - j_0) \cdot (j - j_1) > 0,\ j_1, j_0 \in Z_y \ or\ s_x = 0 \ or\ s_y = 0 \tag{3.6}$$

Scope of illustration: In the general FF-SA problem, the minimum area and width constraints for each choice may vary. For example, in agricultural spatial allocation, several land cover types (e.g., alfalfa) may not require large-size farm equipment operation while the others (e.g., corn) still require it. Thus, α and β can be varied for each choice. Since the proposed HFE algorithm can be trivially generalized to the case where different α and β are used for different choices, for simplicity and clearance of illustration, we assume α and β are the same for each choice in later sections to avoid overloaded details.

3.3 Challenges

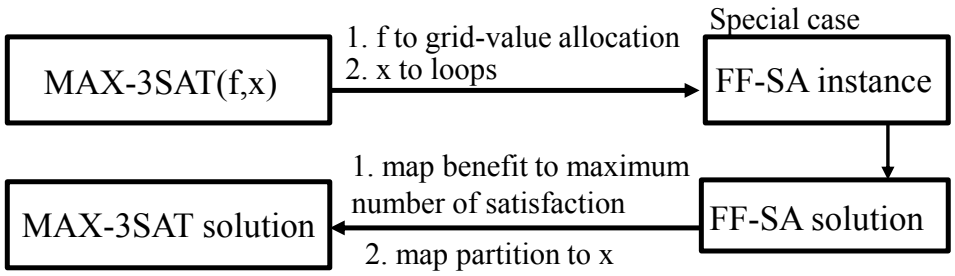


Figure 3.1: Reduction graph.

The FF-SA problem is APX-hard, and the APX-hardness implies NP-hardness. The hardness proof is based on reduction from the MAX-3-Satisfiability (MAX-3SAT) problem [99]. We show FF-SA is more general than the MAX-3SAT problem, which is APX-hard and does not admit a polynomial-time approximation scheme (PTAS). We propose an extension of the proof in [100], which shows 3-Satisfiability (3SAT, a variant of MAX-3SAT) problem is a special case of box packing (BOX-PACK). In general, it was shown that for any 3SAT instance, a box packing instance can be constructed with a special design of box-allocation in polynomial-time, and the optimal solution of box packing can be converted to the solution of the 3SAT problem in polynomial time. We generalize the proof in [100] to show that MAX-3SAT is a special case of FF-SA. The general reduction graph is shown in Fig. 3.1, and the detailed proof is provided in Sec. 3.3.1.

3.3.1 APX-hardness

We start the proof with the following two definitions.

Definition 2 (3SAT and MAX-3SAT)**.** *Given a Boolean formula:* $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_4 \vee x_5) \wedge (\neg x_5 \vee x_6 \neg x_7) \wedge ...$*, which is composed by* m *clauses where each clause*

has 3 binary variables, ***3SAT*** *determines whether there exists a 0-1 assignment of all variables such that the output of the Boolean formula is 1 (satisfiable).* ***MAX-3SAT*** *is a variation of 3SAT. Instead of determining whether the formula is satisfiable, MAX-3SAT finds a 0-1 variable assignment to maximize the number of satisfied clauses.*

Definition 3 (BOX-PACK and MAX-BOX-PACK)**.** *Given a set S of axis-parallel square boxes with integer height h_i, width w_i and coordinates (locations fixed),* ***BOX-PACK*** *determines whether there exists a subset of B non-overlapping boxes in S.* ***MAX-BOX-PACK****, instead of determining this existence, finds the subset that has the maximum number of non-overlapping boxes.*

The proof in [100] reduces 3SAT to BOX-PACK. It was shown given a 3SAT instance, a specific two-dimensional allocation of boxes can be constructed in polynomial time in BOX-PACK, and 3SAT can be satisfied (return 1) if and only if a subset of B non-overlapping boxes can be found. Every unsatisfied clause in 3SAT will cause a deduction of 1 from B. The proof needs to be extended in two ways to reduce MAX-3SAT to FF-SA: (1) extend from "3SAT $\Longrightarrow$ BOX-PACK" to "MAX-3SAT $\Longrightarrow$ MAX-BOX-PACK"; and (2) re-formulate the box-allocation in MAX-BOX-PACK to the grid representation of FF-SA by assigning specific grid cell values.

Polynomial-time reduction from BOX-PACK to MAX-BOX-PACK: (1) If BOX-PACK can be determined in polynomial time, then we show there exists a simple algorithm to find the maximum number B^* in MAX-BOX-PACK. Suppose the total number of boxes in the instance is N. Then the algorithm can simply enumerate through all integers in $[1, N]$ using BOX-PACK and find B^* in polynomial time. (2) If MAX-BOX-PACK can be solved in polynomial time, then for any number B in BOX-PACK, we can compare it to B^* and get the decision in polynomial time. Thus, based on (1) and (2), BOX-PACK and MAX-BOX-PACK can be mutually transformed in polynomial time.

MAX-3SAT is a special case of MAX-BOX-PACK: We show MAX-3-SAT can be reduced to MAX-BOX-PACK in polynomial time. So far we have shown 3-SAT can be reduced to MAX-BOX-PACK. Suppose we have a 3SAT instance I_{3SAT} and a corresponding BOX-PACK instance I_{BP}, and B is the largest number of non-overlapping boxes achievable in I_{BP} when I_{3SAT} is satisfiable. [100] shows that every unsatisfied clause in I_{3SAT} is equivalent to a deduction of one in B of I_{BP}. Since for I_{3SAT} we can create a corresponding MAX-BOX-PACK instance I_{MBP} with exactly the same input (box-allocation) as I_{BP}. Let B^* denote the maximum number of non-overlapping boxes achievable in I_{MBP}, and $B^* = B$ when I_{3SAT} is satisfiable. Similarly, we construct a MAX-3SAT instance I_{M3SAT} using the same input of I_{3SAT}. Given B, B^*, and m (total number of clauses in I_{M3SAT}), the maximum number of satisfied clauses in I_{M3SAT} can be computed as $M^* = m - (B - B^*)$. Thus, I_{MBP} can be constructed in polynomial time given I_{M3SAT}, and the solution M^* of I_{M3SAT} can be achieved in polynomial time given B^*. **MAX-3SAT is a special case of FF-SA:** To prove FF-SA is APX-hard, we use the above-mentioned special case (specific input box-allocation) of MAX-BOX-PACK to build the corresponding instance of FF-SA. First, we reformulate the special case of box-allocation into a grid representation. Since the coordinates of boxes in MAX-BOX-PACK are all integers, it is straightforward to map the boxes to a grid with cell-size 1. Next, we enrich all the grid cells with values (e.g., benefits) to create the corresponding instance of FF-SA. The goal is to make sure that for the special case of box-allocation, the optimal solution of FF-SA gives the optimal solution of MAX-BOX-PACK in polynomial time.

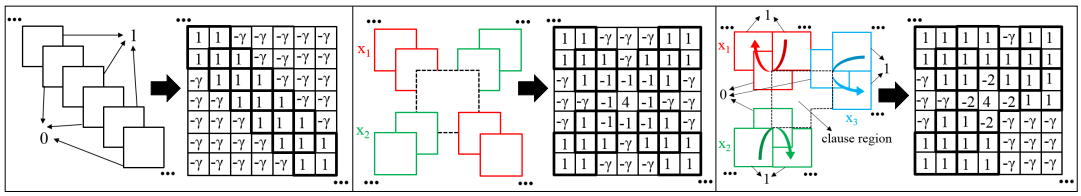
Overview of the construction of box-allocation [100]: Generally, for each variable in MAX-3SAT, there is a loop of equal-size boxes with side-length 2 and the number of boxes is even. Half of the boxes are labeled "1" and the others are "0", where the boxes representing "1" and boxes representing "0" are interleaving (Fig. 3.2(a)), so

that each box with "0" (resp. "1") intersects with two boxes with "1" (resp. "0"). Given this allocation, the largest subset of non-overlapping boxes for each loop is either all boxes with "1" or all with "0", which corresponds to the binary choice of each variable in MAX-3SAT. For a given clause with three variables, there will be a clause region, where the three box-loops corresponding to the three variables approach each other. The space available in the clause region is determined by the 0-1 choices made at the three loops. The allocation is designed in a way that if any loop chooses "1" boxes, then there will be enough space to include one and only one more box in the clause region into the solution (a subset of all boxes) without overlapping any other boxes. Given a solution containing a subset of boxes in MAX-BOX-PACK, the number of satisfied clauses in MAX-3SAT can be inferred by counting how many boxes are added from all the clause regions. [100] shows it is sufficient to discuss three general cases appearing in the box-allocation. To avoid heavy redundancy, we will focus on the three cases without proving how exactly they translates to MAX-3SAT, as this omitted proof is in [100].

In the following discussion of FF-SA instance I_{FF}, (1) we relax the cost constraint by choosing a $+\infty$ budget; (2) there are only two choices for each grid cell, namely "A" and "B". A's benefit values are shown as the grid cell values, and B's benefit values are always 0; and (3) for choice A, the minimum area is 4 and minimum width is 2; for choice B, the minimum area and width are 1 (discussed in scope, Sec. 4.2).

Case 1: A fragment of a single loop and this fragment does not intersect with any other loop (Fig. 3.2(a), left). It is easy to see we can either choose all boxes with "1" or all with "0" for a maximum non-overlapping subset. To achieve the same result in FF-SA, a benefit-value assignment is proposed in Fig. 3.2(a) (right), where γ denotes a very large positive value (e.g., 9999). Since tile with choice "A" cannot overlap and must satisfy minimum area 4 and width 2, the optimal solution must be choice "A" on either all 2×2 grid boxes corresponding to "1" or "0", which is the same as MAX-BOX-PACK.

The rest of the cells will just choose choice "B" with 0 benefit.



(a) Case 1 (b) Case 2 (c) Case 3

Figure 3.2: Three cases of box-allocation: from MAX-BOX-PACK to FF-SA. (best in color)

Case 2: An intersection of two different loops (Fig. 3.2(b), left). An intersection is a special case in this box-allocation. A square region (dashed) of 3×3 is added to the intersection, which contains 4 mutually overlapping 2×2-boxes (not drawn) and the four boxes do not belong to either loop. This special modeling guarantees that whatever combinations of "1" and "0" x_1 and x_2 (Fig. 3.2(b)) choose, we can always add one and only one 2×2-box from the dashed region into the optimal non-overlapping subset. Thus, the intersection has no effect on the decision in choosing the optimal subset. A corresponding benefit-value assignment is proposed in Fig. 3.2(b)(right). For both loops, choosing boxes with either "1" or "0" will include one and only one cell with value of -1 at the intersection. Same as MAX-BOX-PACK, it does not matter whether "0" or "1" boxes are chosen by the two loops to choose choice "A", since we can always add one and only one 2×2 box with choice "A" at the intersection and increase benefit by 1. The cells left in the intersecting region will choose choice "B" (0 benefit value).

Case 3: This case is the only one that affects the final optimal solution of MAX-BOX-PACK and FF-SA. As shown in Fig. 3.2(c), there are three loops of boxes intersecting one dashed region at their turns. This dashed region is called a "clause region". The dashed region has an up-side-down "L" shape and it has three mutually overlapping 2×2-boxes. Each "clause region" in the box allocation corresponds to a clause in the

MAX-3SAT problem. The loops are constructed such that only a "0" box intersects the "clause region". In the clause region, if there exists at least one loop that chooses boxes with "1", then we can add one and only one more 2×2-box to the final subset. If none of the three loops choose boxes with "1", then we cannot add any of the 2×2-box due to overlaps. Thus, for the MAX-BOX-PACK problem, finding a maximum non-overlapping subset is equivalent to finding the maximum number of satisfied clauses in MAX-3SAT. For FF-SA instance I_{FF}, a benefit-assignment is given in Fig. 3.2(c) (right). Similarly, any cell with negative γ is prohibitive for choice "A" and needs to be assigned with choice "B". As with MAX-BOX-PACK, if at least one of the three loops chooses the "1" boxes with choice of "A", then we can add one and only one more 2×2 "A" box into the FF-SA solution. Due to the spatial constraints, the rest of the cells in the clause region must choose "B". In addition, we cannot allocate a 3×2 tile in this region because it will cause a decrease in total benefit.

Thus, with the optimal solution of I_{FF}, all 2×2 boxes with choice "A" represent exactly the optimal subset of boxes chosen in MAX-BOX-PACK instance I_{MBP}, which then gives the optimal solution of the MAX-3SAT instance I_{M3SAT} in polynomial time.

3.4 Related Work and Limitations

Without spatial constraints, there is no need to define the tiles and the spatial allocation problem becomes a 0-1 Multiple-Choice Knapsack Problem (MCKP) [101]. In MCKP formulation, the spatial relationship among grid cells is ignored and each grid cell is considered as an independent variable. There are multiple choices for each variable and each choice has a benefit and cost value. The optimization goal in MCKP is to maximize the benefit under a cost constraint. The 0-1 integer programming problem in MCKP is a well studied problem [101] and can be solved by dynamic programming with a large space cost. However, without modeling of spatial constraints, the allocation results often

show large amount of fragmentation, which prohibits its use in domain applications.

In computational geometry, rectangular partitioning problems have been studied [102, 103]. These problems (e.g., R-TILE, R-PACK) only have one fixed choice for each grid cell and there are no multiple choices to choose from. Thus, R-TILE and R-PACK problem do not concern with spatial contiguity issue and fragmentation is not modeled since there are no different choices to make for adjacent grid cells. The techniques in R-TILE and R-PACK were based on this single-choice property, and thus cannot be trivially applied to FF-SA. The spatially-constrained integer programming formulation of FF-SA can only be exactly solved for very small size problems (e.g., a 10 by 10 grid) given its computational challenge (e.g., APX-hardness result in Sec. 3.3). In this formulation, denote the number of grid cells as N, and the number of choices as d. Then the length of list Z is in $O(N^2 \cdot d)$, and the non-linear pair-wise constraints needed is $O((N^2 \cdot d)^2)$. This makes a 500 by 500 grid with 5 choices require at least 3.125×10^{11} elements (decisions) in Z and 9×10^{22} non-linear pair-wise constraints. [10] proposes a dynamic-growth tiling framework (DGTF) to find a heuristic solution of spatial allocation problem that honors spatial constraints. DGTF focuses on generating a single tiling scheme that satisfy minimum area and width constraints. It has a large potential search space but the algorithm only enumerates one complete tiling scheme within the search space. Thus, it has a very limited enumeration space. In addition, the tile-level choice-assignment phase in [10] is based on heuristic local-search. In this chapter, we targets improving solution quality of FF-SA using a significantly larger enumeration space of tiling schemes, and an exact global optimizer for the tile-level choice assignment phase.

3.5 HFE: Hierarchical Fragmentation Eliminator

The APX-hardness result encourages the design of heuristic algorithms. We propose a Hierarchical Fragmentation Eliminator (HFE), to solve FF-SA with two-phases: (1) space tiling and (2) choice-assignment on tiles. Comparing to [10], the new approach aims to significantly increase the volume of enumeration space of solutions to improve the solution quality of FF-SA.

Search Space vs. Enumeration Space. In general, there exists three types of space tiling frameworks [10, 102], namely 1) arbitrary; 2) hierarchical; and 3) $p \times q$. Arbitrary tiling framework considers all possible tiling schemes. Hierarchical tiling framework uses a straight-line to partition the current rectangle into two sub-rectangles at each step. A $p \times q$ tiling framework considers all schemes with p rows and q columns. A tiling algorithm generates tiling schemes based on these frameworks. We define the **search space** of a tiling framework as the space that contains all possible tiling schemes out of the framework. In contrast, **enumeration space** contains all tiling schemes that are actually enumerated by a tiling algorithm. Since arbitrary tiling framework contains all possible tiling schemes, it has the largest search space. Hierarchical tiling framework includes tiling schemes following the hierarchical bi-partition structure, so its search space is a subset of that of arbitrary tiling framework. $p \times q$ tiling framework is a special case of hierarchical tiling framework, which means its search space is a smaller subset. Although the tiling algorithm proposed in [10] considers a search space defined by arbitrary tiling scheme, it has a very limited enumeration space containing only a single complete tiling scheme. The number of tiling schemes covered by the new HFE algorithm is exponential to the number of grid cells in the input. HFE aims to use this enlarged enumeration space to explore new opportunities to improve solution quality.

3.5.1 Phase-1: Space Tiling.

In phase-1, we propose a Hierarchical Fragmentation Eliminator (HFE) to find a tiling scheme of FF-SA. As discussed in Sec. 3.4, spatial allocation (SA) can be solved using conventional integer programming (IP) formulation without spatial constraints, and this approach is denoted as "IP-SA". IP-SA solution S^*_{IP} is able to maximize benefit under a given cost constraint but contains fragmentation. HFE uses the IP-SA solution S^*_{IP} as an input, and finds a hierarchical tiling scheme to eliminate the land fragmentation in S^*_{IP} with minimum number of choice-changes on grid cells.

The solution quality of IP-SA solution S^*_{IP} can be considered as an upper bound on FF-SA solution S^*_{FF} because the spatial constraints are relaxed. For each grid cell, consider the choice made by S^*_{IP} as the optimal choice, adding spatial constraints will force some non-optimal choices in order to maintain the minimum tile area and width.

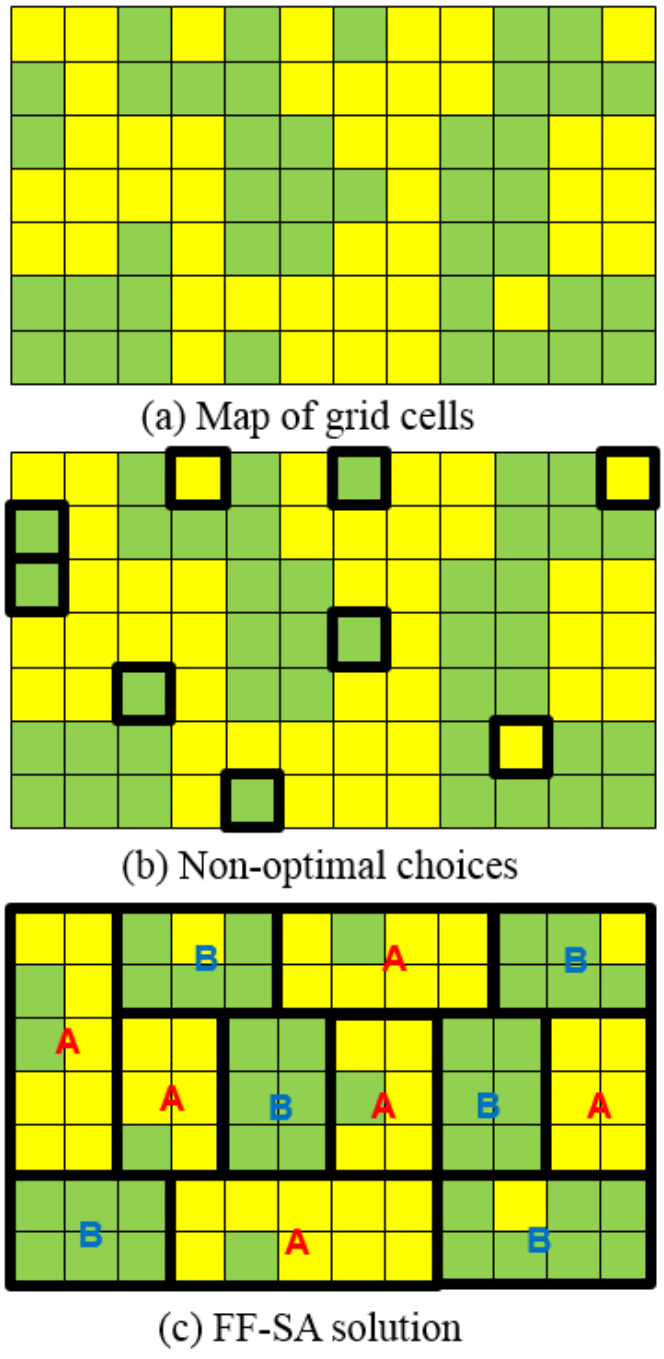


Figure 3.3: A toy example of the spatial allocation problem (best in color).

Table 3.1: Benefit and cost values

Grid cell	Choice A	Choice B
Yellow	**b=2, c=1**	b=1, c=2
Green	b=1, c=2	**b=2, c=1**

*b is benefit, c is cost

Fig. 3.3 shows a toy example of the spatial allocation problem to illustrate the intuition of HFE. In the example, the FF-SA problem aims to maximize the benefit under a cost constraint. There are 84 grid cells and 2 choices "A" and "B" for each cell. For simplicity of demonstration, the cells are colored by green and yellow (to represent spatial variation), and the grid cells of the same color have the same benefit and cost values for each choice. The values are shown in Table 3.1. In this simplified example, we can see "A" is an optimal (dominant) choice for all yellow grid cells since it has a higher benefit of 2 and lower cost of 1; for green cells, "B" is an optimal choice. Suppose the cost constraint is 100. Without any spatial constraints, the optimal integer programming solution S^*_{IP} will always choose "A" for all yellow grid cells and "B" for green, and this gives a benefit of 168 and cost of 84. If we choose a non-optimal choice at a grid cell, the total benefit will decrease by 1 and cost will increase by 1. Suppose we impose two **spatial constraints** to construct the fragmentation-free spatial allocation (FF-SA) problem: minimum tile area 6 and minimum tile width 2 (unit: grid cell). With the constraints, we have to make non-optimal choices on some grid cells in order to satisfy the spatial constraints (in other words, avoid fragmentation). Here "non-optimal" means non-optimal in the IP-SA solution S^*_{IP}. For example, some grid cells are highlighted in Fig. 3.3(b). For each of the highlighted grid cells, we have to make non-optimal choices either on itself or on its neighbor; otherwise it is not possible to

have a valid tile containing it since each tile must have a homogeneous choice. Thus, an optimal solution of FF-SA at least makes non-optimal choices at 9 grid cells. Based on this observation, we can see Fig. 3.3(c) gives an optimal FF-SA solution S^*_{FF} with a benefit of 159 and cost of 93.

Minimization problem of HFE. Within the search space containing all hierarchical tiling schemes, the newly proposed HFE finds a tiling scheme that can globally minimize the total number of choice-changes needed on grid cells to eliminate the fragmentation in the IP-SA solution S^*_{IP} (e.g., Fig. 3.3(b) and (c): 9 changes needed).

The total number of hierarchical tiling schemes can be exponential to the number of grid cells (e.g., when minimum area and width constraints are small). To avoid a brute-force search of all schemes, dynamic programming is used in [102] to minimize the total number of tiles in a hierarchical tiling scheme under a constraint on heft-function values. We show that the minimization problem of HFE also processes the two key properties needed for dynamic programming, namely optimal substructure property and overlapping sub-problems property.

Optimal substructure property: At each level, the hierarchical tiling framework partitions each of the rectangles at this level into two sub-rectangles using a straight line. Denote the range of each rectangle as (i_0, j_0, i_1, j_1), where (i_0, j_0) is top-left grid cell and (i_1, j_1) is the bottom-right. Denote the minimum number of choice-changes needed to eliminate fragmentation in range (i_0, j_0, i_1, j_1) as $M(i_0, j_0, i_1, j_1)$, the minimum tile area in FF-SA as α, and the minimum width as β. Since there are two directions to split a rectangular region into two sub-regions, we have:

$$M(i_0, j_0, i_1, j_1) = \min(M_{hor}, M_{ver}) \tag{3.7}$$

$$M_{hor} = \min_{x_{min} \le x \le x_{max}} [M(i_0, j_0, x, j_1) + M(x+1, j_0, i_1, j_1)] \tag{3.8}$$

$$\begin{cases} x_{min} = \max(i_0 + \beta - 1, i_0 + \lceil \frac{\alpha}{j_1 - j_0 + 1} \rceil - 1) \\ x_{max} = \min(i_1 - \beta, i_1 - \lceil \frac{\alpha}{j_1 - j_0 + 1} \rceil) \end{cases} \tag{3.9}$$

$$M_{ver} = \min_{y_{min} \le y \le y_{max}} [M(i_0, j_0, i_1, y) + M(i_0, y + 1, i_1, j_1)] \tag{3.10}$$

$$\begin{cases} y_{min} = \max(j_0 + \beta - 1, j_0 + \lceil \frac{\alpha}{i_1 - i_0 + 1} \rceil - 1) \\ y_{max} = \min(j_1 - \beta, j_1 - \lceil \frac{\alpha}{i_1 - i_0 + 1} \rceil) \end{cases} \tag{3.11}$$

In Eq. 3.7, the minimum of $M(i_0, j_0, i_1, j_1)$ is achieved either with a horizontal split or vertical split. Since the minimum number of choice-changes needed in each sub-rectangle is independent from the other, the minimum of a split is a direct sum of the minimums of the two sub-rectangles as shown in Eq. 3.8 and Eq. 3.10. This shows the optimal substructure property of the minimization problem in HFE, that is, the solution of the original problem can be efficiently constructed using solutions of the sub-problems. In this case, the construction takes $O(m + n)$ time, where m, n are the number of rows and columns. The feasible locations of a horizontal and vertical split are defined by $[x_{min}, x_{max}]$ and $[y_{min}, y_{max}]$, respectively. The ranges guarantee that the two sub-rectangles generated by the split satisfy the minimum area α and width β.

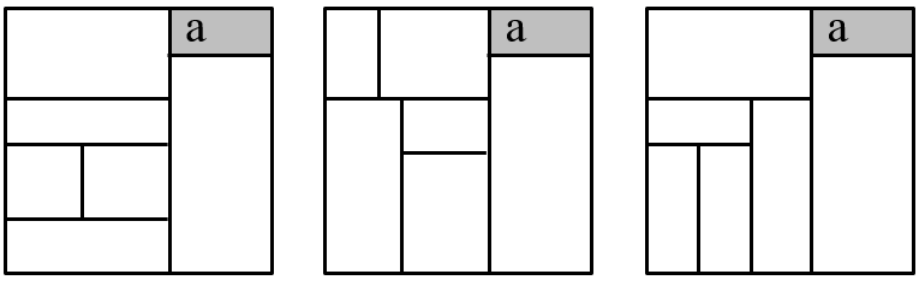


Figure 3.4: Region shared by schemes.

Overlapping sub-problems property: A rectangular region (i_0, j_0, i_1, j_1) in the study area can be heavily shared by many hierarchical tiling schemes. An example is

shown in Fig. 3.4, where the region "a" shows up in all three tiling schemes. In the minimization problem of HFE, the minimum achieved in a region will be re-used by many tiling schemes. HFE uses a **four dimensional storage matrix** R to save the minimums obtained in the sub-problems, and each of the four dimensions corresponds to a coordinate in (i_0, j_0, i_1, j_1) which defines the rectangular region of the sub-problem. During hierarchical tiling, when the minimum of a rectangular region (i_0, j_0, i_1, j_1) is needed, HFE queries $R(i_0, j_0, i_1, j_1)$ for the minimum. If $R(i_0, j_0, i_1, j_1)$ is null, the minimum will be computed based on hierarchical tiling according to Eqs. 3.7-3.11; otherwise the value of $R(i_0, j_0, i_1, j_1)$ will be directly used.

The pseudo-code of HFE consists of two recursive functions: (1) `find_min`: finds the minimum for each rectangular region within the study area to complete matrix R (Alg. 4); (2) `find_tile`: uses the computed matrix R to find all tiles belonging to the optimal solution (Alg. 5). The reason to split the work into two functions is to avoid extra storage needed to save all split points and directions during hierarchical tiling. It is also worth-noting that the process in Alg. 4 is different than a k-d tree, which is a one-way process without recursive functions. In addition, in line 17-19 of Alg. 4, the minimum is computed directly since the rectangular region given is not splittable under the spatial constraints. In this non-splittable case, no sub-problems can be further generated and we need to directly compute the minimum number of choice-changes needed. As defined in Sec. 3.2, all grid cells in a tile must share the same choice in a FF-SA solution. Thus, the minimum is achieved by assigning all grid cells with the majority choice in S^*_{IP} in this region, and the minimum value is the number of grid cells in the region that are not originally assigned with this majority choice.

To find the optimal hierarchical tiling scheme for the minimization problem of HFE, Alg. 4 and Alg. 5 are executed sequentially with inputs $(i_0, j_0, i_1, j_1) = (1, 1, i_{max}, j_{max})$, IP-SA solution S^*_{IP}, minimum area α, minimum width β, R initialized with all NULL

values and an empty list *list* (Alg. 5 only). The output is a complete list *list* containing all tiles in the final tiling scheme.

3.5.2 Phase-2: Choice Assignment

Phase-1 of FF-SA gives an initial solution for spatial allocation. It also finalizes the tiling scheme of FF-SA in the proposed algorithm. A fixed tiling scheme simplifies the FF-SA problem since the only task left is to find an optimal choice assignment.

Preprocessing: This step combines all grid cells in each tile into one single variable. Given the spatial range of each tile, the benefit value and cost value of the combined variable can be computed for each choice by summing up all benefit or cost values of the grid cells inside. **Optimization:** Since the tiling scheme from HFE in phase-1 does not contain fragmentation, we just need to optimize the choices on each tile to maximize the total benefit under the cost constraint for the given tiling scheme. This problem's structure is the same as IP-SA (integer programming formulation without spatial constraints). As discussed in Sec. 3.4, the 0-1 integer programming problem in IP-SA is a well-studied problem and can be formulated into 0-1 multiple-choice knapsack problem [104]. This chapter does not attempt to improve the performance of conventional integer programming solvers, and thus we use standard CPLEX solvers [105] in phase-2 to optimize the choice-assignment.

3.5.3 Algorithm Acceleration

The focus of this section is to reduce the time complexity on computing the minimum of a non-splittable tile (i_0, j_0, i_1, j_1) (line 18 in Alg. 4). The goal is to count the minimum number of grid cells that need a choice-change to remove the fragmentation and get homogeneous choice assignment inside the tile. This minimum number *minRect* is the number of grid cells that are not assigned with the majority choice in this tile (Sec.

3.5.1).

Naive algorithm: The naive algorithm performs a single scan of all grid cells in this region, and counts the number grid cells of each choice. The maximum count $maxRect$ can be tracked and updated in the process. After the scan, the minimum $minRect$ is achieved by $minRect = (i_1 - i_0 + 1) * (j_1 - j_0 + 1) - maxRect$. The time complexity of the naive algorithm is $O(r)$, where r is the number of grid cells in the tile.

Accelerated algorithm: In this algorithm, an integral image [106] of **the whole** S^*_{IP} is pre-computed prior to the start of Alg. 4 to reduce the time complexity. An IP-SA solution S^*_{IP} can be considered a two-dimensional image $I^{m\times n}$ where the value on each pixel is a choice ID assigned on the corresponding grid cell (each grid cell becomes a pixel here). Suppose there are d choices in total. The accelerated algorithm first converts the two-dimensional image into a **multi-layer representation** with dimensions $m \times n \times d$, where d is the number of layers and each layer corresponds to a different choice. Each two-dimensional layer is a binary image: for layer k, if the k^{th} choice is assigned to the pixel, the value is set to 1; otherwise 0. Then, an integral image $I^{m\times n}_{int}$ is computed for each layer. An **integral image** is defined as: Given an image I, its integral image I_{int} has the same dimension, and each pixel value $I_{int}(p,q) = \sum_{i=1}^{p}\sum_{j=1}^{q} I(i,j)$.

Integral image I_{int} can be constructed with two linear scans of the original image I. The first linear scan is performed row-wise. For row i, each element is sequentially updated as: $I_{int}(i,1) = I(i,1)$, and $I_{int}(i,j) = I_{int}(i,j-1)+I(i,j)$, $j = 2,...,n_{col}$. Thus, after the row-wise linear scan, each $I_{int}(i,j) = \sum_{j=1}^{n} I(i,j),\ \forall i$. To get the final integral image, a similar column-wise linear scan is performed. For column j, each element is sequentially updated as: $I_{int}(1,j) = I(1,j)$, and $I_{int}(i,j) = I_{int}(i-1,j) + I(i,j)$, $i = 2,...,m_{row}$. Using I_{int}, the sum of values in a rectangular region (i_0, j_0, i_1, j_1) of the original image I can be computed by $\sum_{i=i_0}^{i_1}\sum_{j=j_0}^{j_1} I(i,j) = I_{int}(i_1,j_1) - I_{int}(i_0-1,j_1) - I_{int}(i_1,j_0-1) + I_{int}(i_0-1,j_0-1)$ [106] in O(1) time.

With each layer of the multi-layer IP-SA solution converted to an integral image, the minimum *minRect* can be computed in O(d) time using Alg. 6, where d is the total number of choices.

3.5.4 Computational Time Analysis

For the computational time analysis, this section mainly focuses on deriving an upper bound for the proposed Minimum Fragmentation Eliminator (MFE) algorithm. The 0-1 integer programming problem in IP-SA is a well studied problem [104] and in general is less difficult than FF-SA since it admits a fully polynomial time approximation scheme whereas FF-SA does not (APX-hard). Thus, in this chapter we assume that IP-SA problem can be solved in reasonable time using the state-of-the-art solvers (e.g., CPLEX [105]), and this is a prerequisite of our proposed HFE algorithm.

Time complexity of HFE: Suppose the input grid of the study area has m rows and n columns ($N = mn$ grid cells), each grid cell has d choices, and the minimum area is α and minimum width is β (unit of α, β is grid cell). In addition, denote the total number of non-splittable rectangular regions as N' (line 17, Alg. 4). N' is determined by m, n, α and β. In order for a region to be non-splittable, both x and y in Eq. 3.8 and Eq. 3.10 must not be feasible. In other words, $x_{min} > x_{max}$ and $y_{min} > y_{max}$. Since the relationship between x_{min} and x_{max} as well as y_{min} and y_{max} are difficult to determine without specific values of (i_0, j_0, i_1, j_1) according to Eq. 3.9 and Eq. 3.11, here we use N' to represent the number of non-splittable regions instead of a close-form expression using m, n, α and β.

Theorem 4. *The upper bound on computational time complexity of HFE (accelerated) is* $O((N^2 - N')(m + n))$. *The upper bound on computational time complexity of HFE (naive) is* $O((N^2 - N')(m + n) + \alpha N')$.

Proof. Since each rectangular region can be uniquely identified by two grid cells (e.g.,

top-left and bottom-right), there are at most N^2 different rectangular regions to be enumerated. Among these rectangles, we need to enumerate split points for $N^2 - N'$ of them. For each splittable rectangular region, there exist at most $m + n$ possible split points. Thus, to get the solution, the total number of queries on the rectangles is at most $(N^2 - N') \times (m+n)$. For each non-splittable rectangle, HFE uses the multi-layer integral image to compute its minimum in $O(d)$ time by enumerating d layers. Thus, the total time complexity of Alg. 4 is upper bounded by $O((N^2 - N')(m+n) + dN')$. In order to use the accelerated algorithm, the multi-layer integral image needs to be computed prior to the execution of Alg. 4. Since only two linear scans of each layer is needed, the multi-layer integral image can be computed in $O(dN)$ time. Thus, the total time complexity for multi-layer image computation and Alg. 4 is $\mathrm{O}((N^2 - N')(m + n) + dN' + dN)$. In FF-SA the total number of choices on each cell is assumed to be much smaller than $(m + n)$ (e.g., 5 vs. 500), so we have $d << (m + n)$. In addition, N is dominated by N^2. Thus, the upper bound is simplified to $O((N^2 - N')(m + n))$. Alg. 5 finds all the tiles using a top-down and one-way search. For each rectangle at a certain level in the hierarchy, we enumerate at most $m + n$ split points. Since all the minimum values *minRect* of the rectangular regions have been computed and stored in R, we only need to check which one of the split point gives that the sum of the minimums from the two sub-rectangles is equal to the minimum of the current rectangle. After one split point satisfies this criterion, there is no need to further check other splits and this split must belong to an optimal tiling scheme. In an optimal tiling scheme, there are at most $\lfloor \frac{N}{\alpha} \rfloor$ tiles given the minimum area constraint. Since the hierarchical tiling framework partitions each rectangular region into two sub-regions at each step, we can consider the final hierarchy as a full binary tree. Thus, given that the tree has at most $\lfloor \frac{N}{\alpha} \rfloor$ leaf nodes, there are at most $2\lfloor \frac{N}{\alpha} \rfloor$ nodes in the full binary tree. Thus, Alg. 5 requires at most $2\lfloor \frac{N}{\alpha} \rfloor (m + n)$ enumerations of split points, and the result of each split can be

evaluated in $O(1)$ time (e.g., line 4 and 13 in Alg. 5). This shows the time complexity of Alg. 5 is bounded by $O(\lfloor \frac{N}{\alpha} \rfloor (m+n))$. Thus, the total time complexity of HFE is $O((N^2 - N' + \lfloor \frac{N}{\alpha} \rfloor)(m+n)) = O((N^2 - N')(m+n))$.

For the naive algorithm, the only difference lies in the computation of minimums for non-splittable rectangular regions. The accelerated algorithm needs $O(d)$ time for the computation whereas the naive algorithm requires $O(\alpha)$ time since it needs to enumerate all grid cells in each non-splittable region. As analyzed in the proof above, the time complexity of the accelerated version can be written as $O((N^2-N')(m+n)+dN')$ where the second term represents the time needed for computing minimums for non-splittable regions. Since d is subsumed by $m+n$, the final time complexity is $O((N^2-N')(m+n))$. In the naive algorithm case, the time is $O((N^2 - N')(m+n) + \alpha N')$. The difference is that α may not be subsumed by $m+n$ and in fact α can be much greater than $m+n$ (e.g., when α is a portion of $N = mn$). Thus, the final time complexity is written as $O((N^2 - N')(m+n) + \alpha N')$. □

Theorem 5. *The space complexity of HFE is $O(N^2)$.*

Proof. In HFE, there are in total N^2 sub-problems as shown in the proof of Thm. 1. Thus, storing the minimums of all sub-problem requires at most $O(N^2)$ space. □

3.6 Validation: Case Study in Agricultural Land Allocation

We evaluate the performance of the proposed HFE algorithm through a case study on agricultural land allocation at the Seven Mile Creek watershed in Minnesota, US. The study area is discretized into a 455×477 grid with 30 meter by 30 meter grid cells. In this case study, the choices are 5 land management practices, namely conventional tillage, conservation tillage (with corn stover), low-fertilizer application, prairie grass

and switch grass. The **benefit** is the amount of sediment reduction in water and the **cost** is the investment needed for changing current land management practices. The current land management practice in the study area is homogeneously conventional tillage.

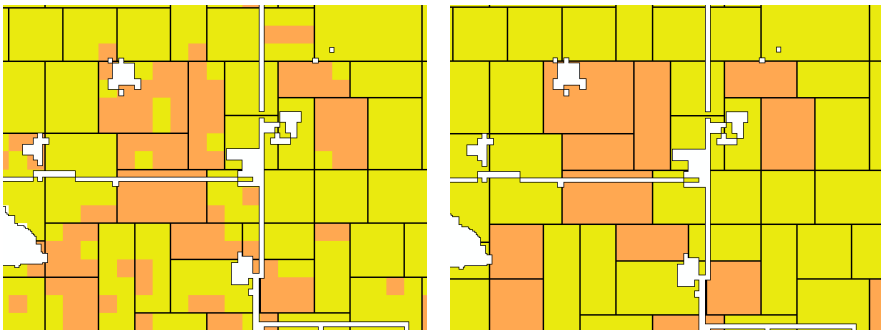


Figure 3.5: Fragmentation elimination.

Since the original shape of the study area is irregular while the rectangular version of FF-SA assumes a rectangular study area, an orthogonal rectangular bounding box (ORBB) of the study area was used to generate the grid. All grid cells outside the original study area in the ORBB are assigned with 0 benefit and 0 cost for all choices so that they do not impact the final solution. In addition, there are some unchangeable landscape in the study area (e.g., roads). The grid cells at these places are also assigned with zero benefit and cost values. In the current work, the minimum size and shape constraints are relaxed at the irregular boundaries of both the study area and unchangeable landscapes.

Fig. 3.5(a) shows the IP-SA solution (integer programming without spatial constraints) at a local zoom-in window overlaid with the tiling scheme generated by the HFE algorithm. In the maps, each color represents a choice of land management practice. There are some regions (white) cut out from the tiling scheme, and those regions

are the unchangeable landscape inside the study area as discussed above. We can see fragmentation in the IP-SA solution, such as irregular shapes and tiny area of patches (e.g., standing alone grid cells). The land fragmentation prohibits efficient use of modern farm equipments (e.g., combine harvesters). The fragmentation is removed by the HFE algorithm with each tile being constrained by a minimum area and width. Based on the hierarchical tiling scheme, the HFE algorithm minimizes the number of choice-changes needed to remove the fragmentation in a IP-SA solution. Fig. 3.5(b) shows the final output of FF-SA at this local window with both a tiling scheme and choice assignment.

3.6.1 Solution Quality Comparison

In this section, we compare the solution quality of the proposed HFE algorithm and the dynamic-growth tiling framework (DGTF) in [10]. The goal of this comparison is to evaluate the impact of tiling schemes on final solution quality (e.g., total benefit). Thus, we use the same choice-assignment algorithm in phase-2 after a tiling scheme is generated from either HFE or DGTF to eliminate the impact of choice-assignment algorithm. Since the original choice-assignment algorithm used for DGTF in [10] is a heuristic algorithm whereas the proposed phase-2 algorithm in this work is an exact algorithm (global optimizer), using the new choice-assignment algorithm for DGTF will strictly improve the solution quality of DGTF in [10].

Experiment setup: We consider two resolutions of the input grid. The first grid is the default grid where the cell-size is 30m. The second grid reduces the resolution to half and has a cell-size of 60m. For each of the grid, we evaluate 5 different combinations of (minimum area, minimum width): (8,2), (50,5), (200,10), (800,20) and (1800,30), where the unit is grid cell. The budget limit used is $100,000 as suggested by domain experts. As a reference, the smallest minimum area used is about the size of two standard American football fields (playing field). **Comparison:** Fig. 3.9 and Fig. 3.10 show

comparisons of solution quality of HFE and DGTF. The vertical axis shows sediment reduction (tons/year) and horizontal axis shows minimum area (grid cell) imposed on tiles. The first trend in Fig. 3.9 and Fig. 3.10 is that the solution quality of both algorithms decrease as the minimum area increases. As the minimum area increases, space becomes more contiguous with larger tile area. However, the algorithms are also forced to make more non-optimal choices (compared to IP-SA solution) at places in order to maintain a higher level of contiguity and regularity. The second trend is that the solution quality of HFE is consistently better than DGTF in the experiment. The result is expected since the enumeration space of HFE is much larger than that of DGTF and it provides potential opportunities to identify better solutions during optimization. The average difference between solution quality of HFE and DGTF is about 51 tons/year in Fig. 3.9 and 64 tons/year in Fig. 3.10. Fig. **??** shows the maps of land allocation achieved by HFE with three different minimum area constraints (unchangeable landscape removed). The grid cell size is 30m (default).

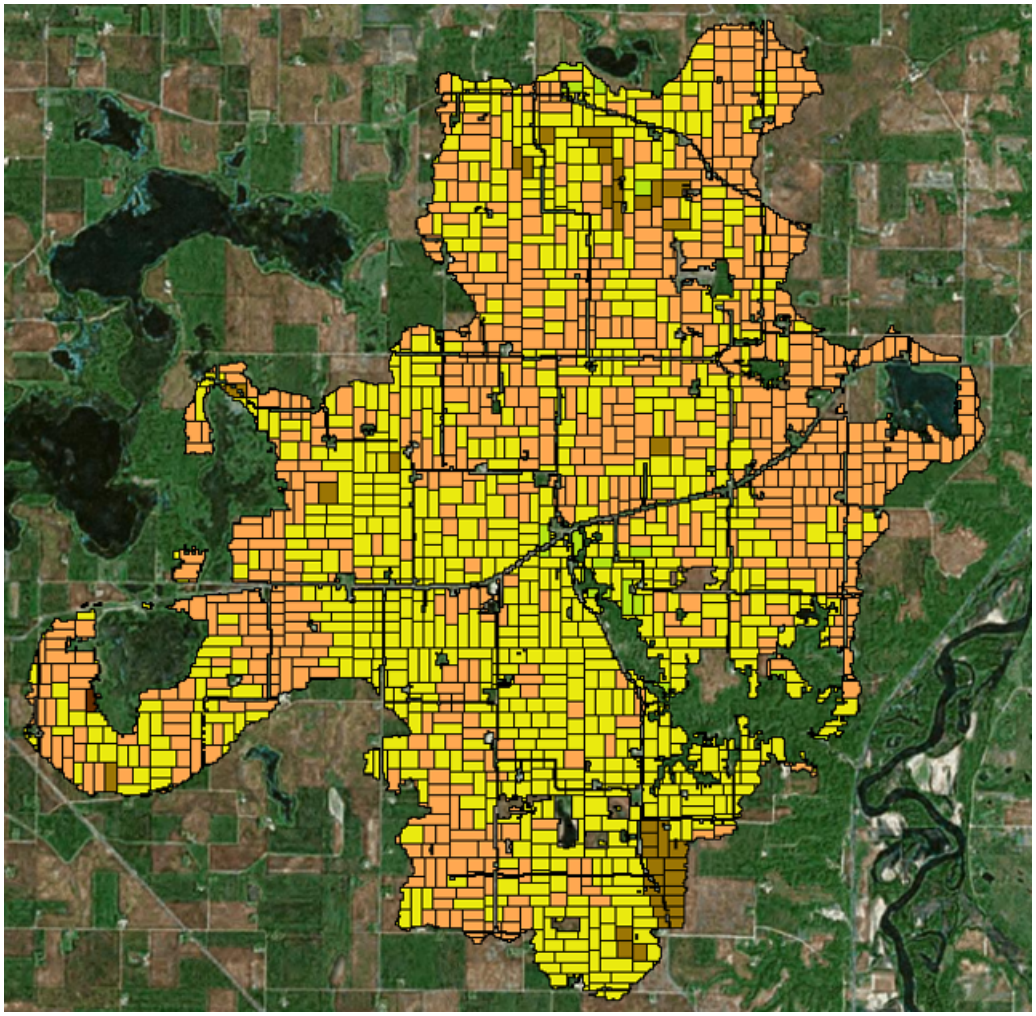


Figure 3.6: Spatial allocation result with spatial constraints: $\alpha = 50$ and $\beta = 5$.

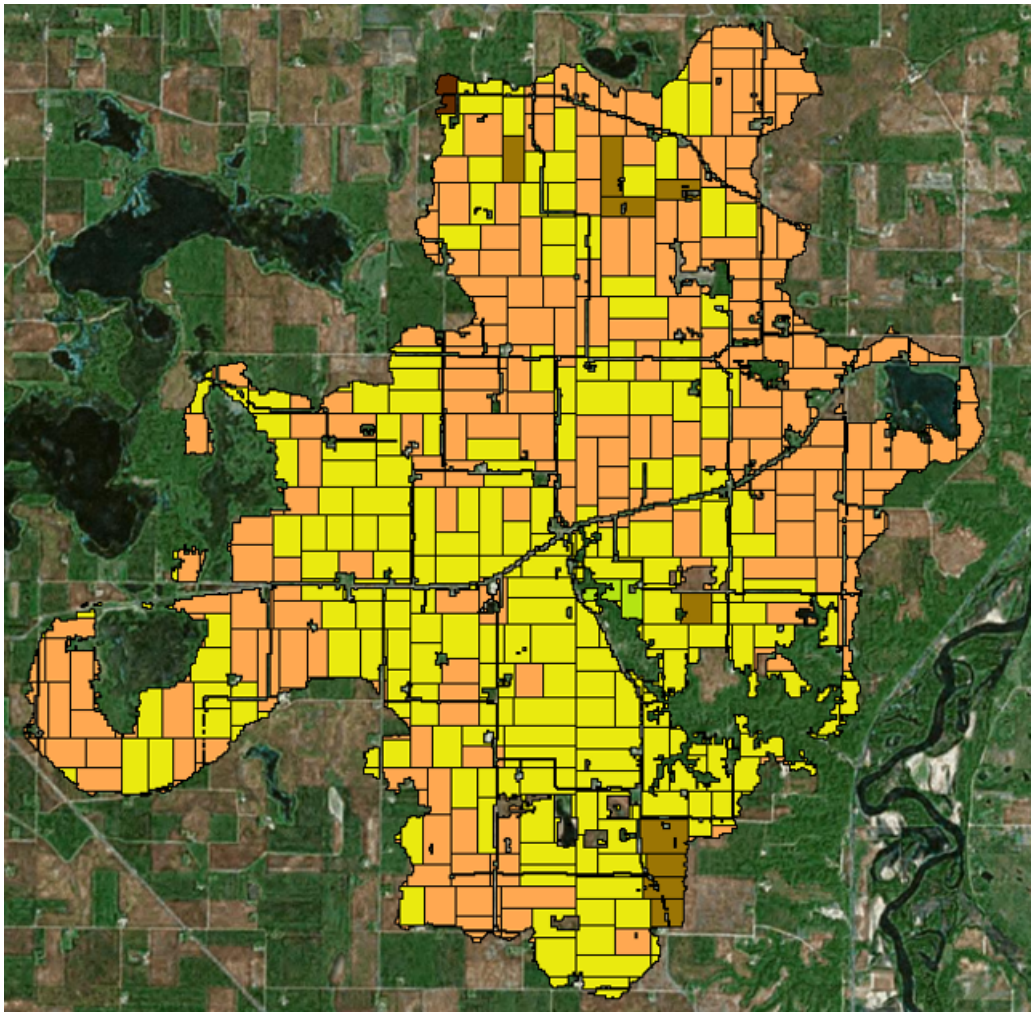


Figure 3.7: Spatial allocation result with spatial constraints: $\alpha = 200$ and $\beta = 10$.

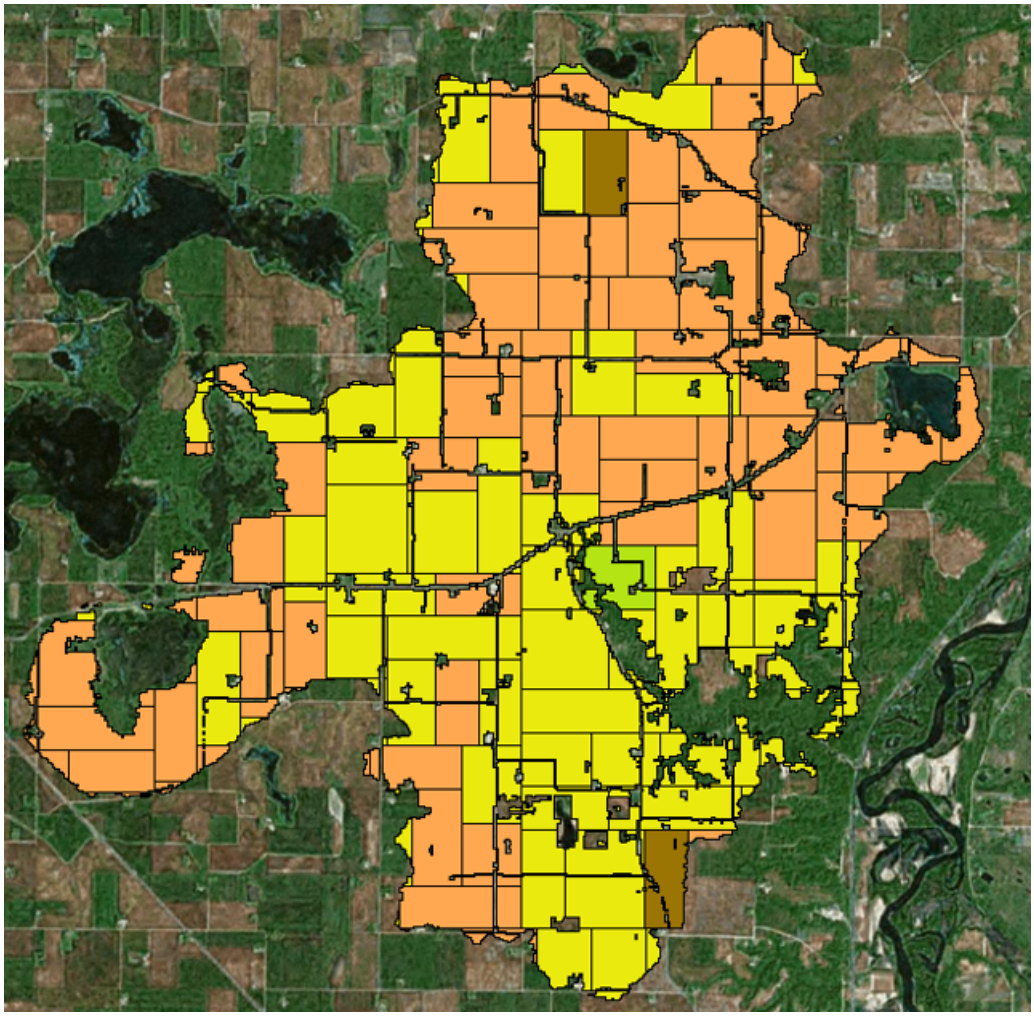


Figure 3.8: Spatial allocation result with spatial constraints: $\alpha = 800$ and $\beta = 20$.

3.6.2 Execution-Time Analysis

Runtime is measured on a 64-bit Windows 8 laptop with Core i7 and 8 GB of RAM. Since large N leads to very expensive time cost for the naive algorithm (Sec. 3.5.4), we used a grid with 100m cell-size to compare the average run-time of the naive and the accelerated algorithm. The chart in Fig. 3.11 shows the performance improvement gained by algorithm acceleration proposed in Sec. 3.5.3. The time is shown in seconds

using a log scale (Y-axis). For the accelerated HFE, the execution time decreases fast as the minimum area α increases, because in general the hierarchical split can stop earlier for larger minimum area α. The performance of the naive algorithm is more complicated. When α is small, the computational time increases sharply as α increases. However, as α becomes larger, the time begins to decrease as α increases. This is because the naive algorithm's time complexity is $O((N^2 - N')(m + n) + \alpha N')$, where α shows up as one of the dominant term (N' is the total number of non-splittable rectangles). When α is small, N' remains small. For example, when $\alpha = 1$, $N' = N$. Then as α increases, N' also temporarily increases. However, when α gets very large, N' becomes small again. For example, an extreme case is when $\alpha = N$, $N' = 1$. Thus, this reflects the increasing and decreasing trends in the naive algorithm's execution time.

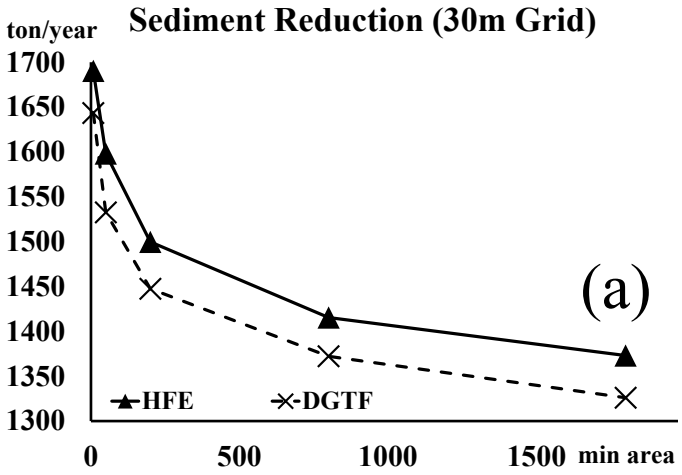


Figure 3.9: Solution quality comparison with 30m-grid.

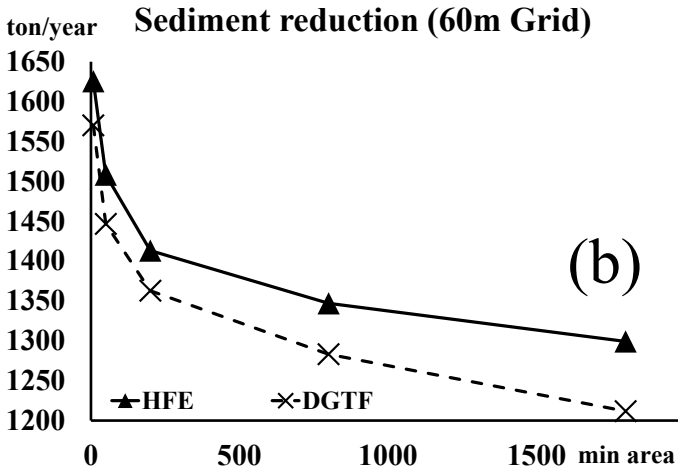


Figure 3.10: Solution quality comparison with 60m-grid.

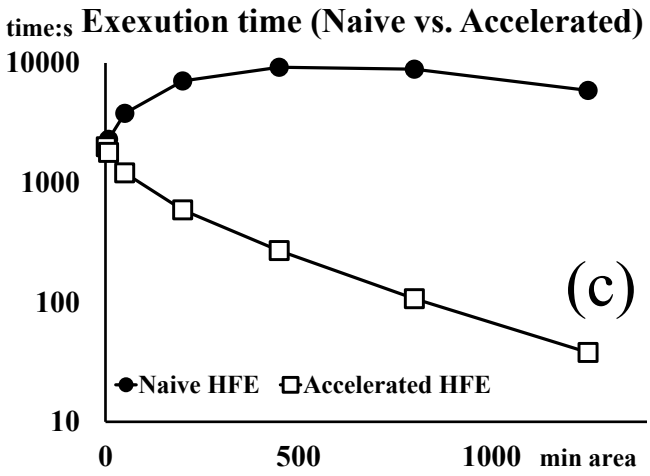


Figure 3.11: Execution time comparison.

3.7 Conclusion and Future Work

We propose a Hierarchical Fragmentation Eliminator (HFE) to solve the Fragmentation-Free Spatial Allocation (FF-SA) problem. FF-SA is important for many societal applications such as agricultural landscape design and urban land-use planning, and it is

computationally challenging (APX-hard as proved). Compared to the related work in optimization and computational geometry, the proposed HFE algorithm addresses the fragmentation-issue while having a significantly larger enumeration space, which helps improve solution quality. The evaluation results show that the HFE algorithm indeed provided higher solution quality compared to existing algorithms and fixed the fragmentation issue. Future work will explore: (1) penalty function formulations of spatial constraints and (2) a variety of regular-shape constraints.

Algorithm 4 find_min$(i_0, j_0, i_1, j_1, S^*_{IP}, \alpha, \beta, R)$

Require:

- rectangular region: (i_0, j_0, i_1, j_1)
- IP-SA solution matrix S^*_{IP}
- minimum area α and width β
- storage matrix R of sub-problem minimums

Initialization: $minRect = +\infty$; $splittable$ =FALSE
if $R(i_0, j_0, i_1, j_1)$! =NULL **then return; end if**
$x_{min} = \max(i_0 + \beta - 1, i_0 + \lceil(\alpha/(j_1 - j_0 + 1))\rceil - 1)$
$x_{max} = \max(i_0 - \beta, i_0 - \lceil(\alpha/(j_1 - j_0 + 1))\rceil)$
for $x = x_{min} : x_{max}$ **do**
    $splittable$ = TRUE
    find_min$(i_0, j_0, x, j_1, S, R, \alpha, \beta)$; find_min$(x + 1, j_0, i_1, j_1, S, R, \alpha, \beta)$
    $minRect = \min(minRect, R(i_0, j_0, x, j_1) + R(x + 1, j_0, i_1, j_1))$
end for
$y_{min} = \max(j_0 + \beta - 1, j_0 + \lceil(\alpha/(i_1 - i_0 + 1))\rceil - 1)$
$y_{max} = \max(j_0 - \beta, j_0 - \lceil(\alpha/(i_1 - i_0 + 1))\rceil)$
for $y = y_{min} : y_{max}$ **do**
    $splittable$ = TRUE
    find_min$(i_0, j_0, i_1, y, S, R, \alpha, \beta)$; find_min$(i_0, y + 1, i_1, j_1, S, R, \alpha, \beta)$
    $minRect = \min(minRect, R(i_0, j_0, i_1, y) + R(i_0, y + 1, i_1, j_1))$
end for
if $splittable$ == FALSE **then**
    $minRect = area(i_0, j_0, i_1, j_1)$ - majority$(S^*_{IP}, i_0, j_0, i_1, j_1)$.count()
end if
$R(i_0, j_0, i_1, j_1) = minRect$

Algorithm 5 find_tile$(i_0, j_0, i_1, j_1, R, \alpha, \beta, list)$

Require:

- rectangular region: (i_0, j_0, i_1, j_1)
- storage matrix R of sub-problem minimums
- minimum area α and width β
- list of tiles found: $list$

1: $x_{min} = \max(i_0 + \beta - 1, i_0 + \lceil(\alpha/(j_1 - j_0 + 1))\rceil - 1)$
2: $x_{max} = \max(i_0 - \beta, i_0 - \lceil(\alpha/(j_1 - j_0 + 1))\rceil)$
3: **for** $x = x_{min} : x_{max}$ **do**
4: **if** $R(i_0, j_0, x, j_1) + R(x + 1, j_0, i_1, j_1) == R(i_0, j_0, i_1, j_1)$ **then**
5: find_tile$(i_0, j_0, x, j_1, R, \alpha, \beta, list)$
6: find_tile$(x + 1, j_0, i_1, j_1, R, \alpha, \beta, list)$
7: **return**
8: **end if**
9: **end for**
10: $y_{min} = \max(j_0 + \beta - 1, j_0 + \lceil(\alpha/(i_1 - i_0 + 1))\rceil - 1)$
11: $y_{max} = \max(j_0 - \beta, j_0 - \lceil(\alpha/(i_1 - i_0 + 1))\rceil)$
12: **for** $y = y_{min} : y_{max}$ **do**
13: **if** $R(i_0, j_0, i_1, y) + R(i_0, y + 1, i_1, j_1) == R(i_0, j_0, i_1, j_1)$ **then**
14: find_tile$(i_0, j_0, i_1, y, R, \alpha, \beta, list)$
15: find_tile$(i_0, y + 1, i_1, j_1, R, \alpha, \beta, list)$
16: **return**
17: **end if**
18: **end for**
19: $list.insert(i_0, j_0, i_1, j_1)$

Algorithm 6 $minRect$=compute_min$(i_0, j_0, i_1, j_1, I_{int}[d])$

Require:

- region: (i_0, j_0, i_1, j_1)
- multi-layer integral image $I_{int}[d]$ (d layers)

1: $maxRect = -\infty$

2: **for** $k = 1 : d$ **do**

3: $\quad count_k = I_{int}[k](i_1, j_1) - I_{int}[k](i_0 - 1, j_1) - I_{int}[k](i_1, j_0 - 1) + I_{int}[k](i_0 - 1, j_0 - 1)$

4: $\quad maxRect = \max(maxRect, count_k)$

5: **end for**

6: $minRect = (i_1 - i_0 + 1) * (j_1 - j_0 + 1) - maxRect$

Chapter 4

A TIMBER Framework for Mining Urban Tree Inventories Using Remote Sensing Datasets

4.1 Introduction

Tree inventories contain critical information for many important societal applications, including resilience of smart cities and communities (e.g., energy infrastructure, green infrastructure), public safety, urban planning, natural resource management, etc.

Trees near electricity power lines pose major threats to energy infrastructure security, especially in the face of climate change. For example, the 2003 Northeast Blackout [107], which was caused by unmanaged trees falling on power lines and subsequent cascade, affected over 50 million people. Such threats will likely worsen in the future due to climate change effects, aging infrastructure, and rapid population growth in cities. In general, fallen or untrimmed trees are responsible for a significant portion of power outages (e.g., about 67% for DTE Energy [108]). Strong winds during severe

storms can knock out trees, snapping power lines and causing blackouts that disturb the operation of hospitals, public transportation and businesses for extended periods. Such events are becoming increasingly frequent in many geographic regions (e.g., across the coasts of Atlantic and Gulf of Mexico). In recent years, hurricanes (e.g., Hermine, Irma and Michael in Florida) have exposed weaknesses in cites' preparation to such catastrophic storms. Fallen trees caused long term electricity outages, road closures, service disruptions and loss of lives. In 2018, the power outage in Tallahassee (FL, USA) caused by broken power lines during Hurricane Michael affected 97% of the city's electric utility customers [109]. In Puerto Rico, a "single" fallen tree cut the main power-line and led to a power blackout for 900,000 customers as well as social outrage [110].

Trees near power lines have also caused many devastating wild fires in recent years. In California, the inability of effectively locating and trimming trees caused a series of deadly fires in 2018, including the Camp Fire, the deadliest wild fire in California history. The fires killed many people and destroyed over 10,000 built structures (e.g., homes) worth billions of dollars [56]. Smoke and unhealthy air that spread to cover over 20 cities caused schools to close for a week and severely hampered people's daily activities.

While trees growing in undesired locations (e.g., near power lines) can pose severe threats, in general they are necessary and invaluable components of our living areas. Trees purify the air, reduce urban heat island effects, create an aesthetically beautiful environment and promote mental health. However, many communities are facing a drastic loss of trees due to the global-spread of pests and disease (e.g., emerald ash borer, pine beetle, dutch elm disease, oak wilt). For example, the emerald ash borer, an invasive insect species, has expanded to 35 US states and killed millions of ash trees [111]. This can lead to severe environmental problems since ash trees cover 20-30% of

treescapes in the majority of urban areas across the US. The cost of locating, treating or removing these ash trees has been estimated to be over 10 billion US dollars in the US alone. Another study found that the tree cover in US has declined in metropolitan areas across 45 states, resulting in an annual net loss of 36 million trees [112]. The same problem also exists in many other geographic areas. For example, ash trees in Europe are also facing extinction [113].

Despite the importance of inventories of individual trees, they rarely exist in most urban areas due to the difficulty of manual collection. For example, it is difficult and time-consuming to manually locate individual trees and measure their canopy sizes (e.g., limited GPS signals under canopies).

We aim to automate the generation of individual tree inventories using high-resolution (e.g., one meter or lower) remote sensing datasets that are publicly available at large-scales. The scope of the present study is to identify the locations and sizes of individual trees in an urban area. The type of remote sensing data that we use is the Normalized Height Model (NHM), which is a single band image whose pixel values represent surface heights. NHMs are LiDAR-derivatives that has been collected and made publicly available at large scales (e.g., state-level or major urban areas across the US).

The tree identification problem is challenging: (1) Trees in urban environments are often mixed with buildings, low-vegetation, lawns, towers, etc; (2) Trees commonly appear in groups with heavy canopy overlaps; and (3) Individual tree inventories are rarely collected (or shared in public) at large scales or in different urban areas, making it difficult to train a generalizable machine learning model that can be applied robustly in different geographical regions.

In NHMs, the canopies of trees are dome-shaped, which makes them similar to mixtures of Guassians. However, Gaussian mixture models (e.g., k-means, expectation-maximization [114]) rely on an input number of clusters, which is unknown in this

problem. These models also do not distinguish tree and non-tree structures (e.g., buildings, towers). In recent years, deep learning models have shown promising results for general computer vision problems as well as urban land-use classification [115, 116]. However, they require a large number of training samples from different geographies, which are not available for the tree detection problem. In addition, deep learning models typically target input images of specific sizes (e.g., 416×416). For remote sensing data, this requires additional space partitioning, which tends to break objects on the boundaries into pieces. Tree detection algorithms have also been studied in the field of remote sensing [117, 118, 119, 120, 121, 122]. These algorithms were mainly designed for landscapes that are mostly homogeneous with few types of trees (e.g., pine tree), and the data sources were specifically collected at very high resolution (e.g., centimeters, hyperspectral). Such datasets are still unavailable at larger scales due to their high costs and limited public availability. Typically these algorithms employ watershed segmentation or clustering to delineate tree canopies. However, the segmentation methods tend to get stuck in small local neighborhoods (e.g., sub-tree levels), and cannot avoid non-tree structures in complex urban environments.

We propose a two-phase TIMBER (Tree Inference by Minimizing Bound-and-band ERrors) framework to identify individual trees in urban environments. The first phase infers the locations and sizes of tree-like structures by optimizing tree-like approximators (e.g., Gaussians) to minimize the difference with tree canopy bounds (i.e., bound errors). The second phase integrates the inputs and outputs from the first phase as well as additional city infrastructure data (e.g., buildings, roads) available for a subset of the study area, and uses them to train a deep convolutional neural network to filter out non-tree results. The deep network predictions are formed by the band values of the input remote sensing data, so we consider this training process as a minimization of band errors. A Core Object REduction (CORE) algorithm is also proposed to improve

the computational efficiency.

Through detailed experiments we show that the proposed TIMBER framework significantly improves precision, recall and F1-scores compared to related work and the CORE algorithm speeds up execution by 1.5x to 2x.

4.2 Problem Definition

4.2.1 Key Concept

A *Normalized height model (NHM)*, also known as a canopy height model [119], is a single-band image (satellite view) whose pixel values represent the height of objects (e.g., buildings, trees) instead of colors. It is a LiDAR-derivative and has been collected at large scales (e.g., many states in the US).

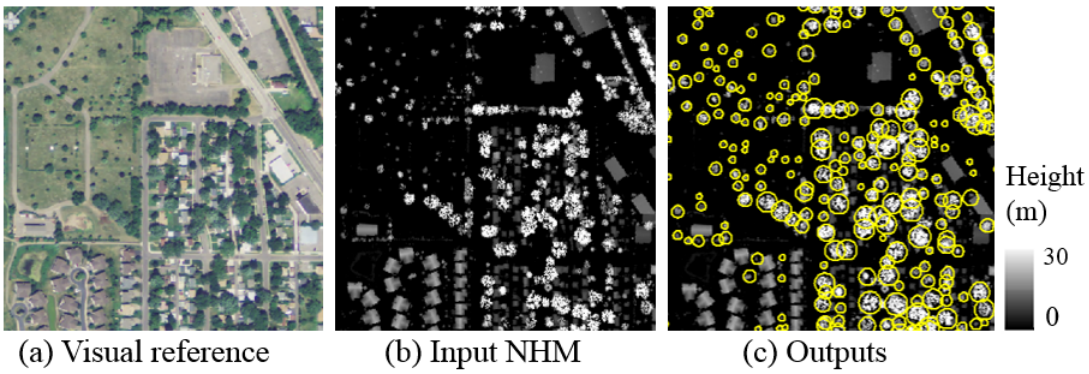


Figure 4.1: Example of input and output. (best in color)

4.2.2 Formal Problem Formulation

Inputs:

– A normalized height model in a spatial domain D;

– Min and max tree sizes, r_{min} and r_{max}, to detect in D;

Outputs: Locations and sizes of trees in D;

Objectives:

– Accuracy of tree detection;

– Computational efficiency;

Constraint: Trees of minimum size r_{min} must be recognizable at the spatial resolution s_{nhm} of the NHM.

Fig. 4.1 shows an example of inputs and outputs, where the shapes of trees are approximated by circles.

4.3 A TIMBER Framework for Tree Identification

TIMBER has two phases. Phase 1 estimates the locations and sizes of tree-like structures using localized optimizations. To remove the non-tree structures, Phase 2 constructs a deep learning filter using a combination of the detections from Phase 1 and city infrastructure datasets (e.g., buildings, roads).

4.3.1 Phase 1: Optimization of Tree Locations and Sizes

In a normalized height model, the structure of a tree is represented by a dome-shaped bound on the top of its canopy, which can be approximated by a mathematical approximator with a varying set of parameters (e.g., Gaussian). We consider the difference between a tree bound and an approximator as the bound error E_b. In Phase 1, TIMBER optimizes a given mathematical approximator to minimize E_b of each tree. Note that Phase 1 yields both trees and tree-like structures (e.g., buildings, towers). The non-trees are filtered out in Phase 2.

Flexible location initializer with local maxima: Trees may have different heights and sizes, but their geometric shapes do share one characteristic, that is, each

tree bound contains a maximum height near the center, which represents the tree-top. These local maxima in normalized height models (NHM) have been used to represent tree locations in forests [118]. We use local maxima, defined as pixels whose value is the largest in a local window, to initialize the rough locations of trees (i.e., center peaks of tree canopies) in an urban environment. Since trees do not have perfect dome-shapes, the bounds of their canopies are not smooth and have small height fluctuations. This often results in multiple local maxima on top of a tree canopy, or makes a local maximum not at the center of a tree. Thus, the local maxima are just rough estimations of actual tree locations.

To achieve better estimations, our optimization formulation considers flexible locations of a local maximum. That is, all locations within its circular neighborhood of radius r_{min} become candidate centers for its corresponding object and the best location will be returned as the center.

Optimization of mathematical approximators: We use mathematical approximators (e.g., [123, 124]) to approximate the dome-shapes of trees in NHMs by minimizing the bound error E_b between the approximator and the actual tree bound. Our optimization method can be generally applied to dome-shaped mathematical surfaces. The general optimization formulation for a single approximator on a single tree is:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu},r} \frac{\|\big(\mathbf{y}(\mathbf{X}) - f_{apx}(\mathbf{X},\boldsymbol{\alpha},\boldsymbol{\mu},r)\big)\cdot \mathbb{1}(\mathbf{X}^T,\boldsymbol{\mu},r)\|_2^2}{\|\mathbb{1}(\mathbf{X}^T,\boldsymbol{\mu},r)\|_1} \tag{4.1}$$

$$\text{s.t. } \|\boldsymbol{\mu} - \boldsymbol{\mu}_{max}\|_2 \leq r_{min} \tag{4.2}$$

$$r \in \{\lceil \frac{r_{min}}{s_{nhm}} \rceil, \lceil \frac{r_{min}}{s_{nhm}} \rceil + 1, ..., \lceil \frac{r_{max}}{s_{nhm}} \rceil\} \tag{4.3}$$

$$\mathbb{1}_i\left(\mathbf{X}^T, \boldsymbol{\mu}, r\right) = \begin{cases} 1, & \text{if } \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2 \leq r \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

where the subscript i indicates the i^{th} row of a matrix or vector; $\mathbf{X} \in \mathbb{Z}_+^{N\times 2}$ is a matrix

containing locations (i.e., row and column IDs in the input NHM) of all pixels, and N is the total number of pixels; $\boldsymbol{\alpha}$ is the parameter vector of an input approximator function $f_{apx}()$; $\boldsymbol{\mu} \in \mathbb{Z}_+^2$ is a vector of the center location (row and column IDs) of the approximator and r is its radius; $\mathbb{1}()$ is an indicator function defined in Eq. (4.4) that returns a vector indicating if a location in $\mathbf{X}$ is within the radius r from the center of the approximator; $\mathbf{y}(\mathbf{X})$ is a vector of the actual height values in the NHM at locations in $\mathbf{X}$; and $\boldsymbol{\mu}_{max}$ is the location of a local maximum on the tree.

The objective function aims to minimize the mean difference between the approximator and the tree bound inside a circular spatial domain of size r. Constraint (4.2) reflects the search distance used in the flexible center formulation. Its limit is set to r_{min} to avoid moving into the center of another tree. Constraint (4.3) defines the set of candidate radii (in pixels) constrained by the input minimum and maximum tree sizes, r_{min} and r_{max}, and the spatial resolution s_{nhm} of the NHM.

To make our discussion more concrete, we illustrate the optimization process with two approximators, i.e., a negative quadratic function and a Gaussian probabilistic function:

Optimization with quadratic surfaces

Negative quadratic functions form dome-shaped mathematical surfaces. Our quadratic approximator f_q is defined as follows:

$$f_q\big((\mathbf{X}_i)^T, \boldsymbol{\alpha}, \boldsymbol{\mu}, r\big) = -\boldsymbol{\alpha}_1 \cdot \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2^2 + \boldsymbol{\alpha}_2 \tag{4.5}$$

where subscript i denotes the i^{th} row of a matrix; $\boldsymbol{\alpha}_1 \in \boldsymbol{\alpha}$ controls the vertical stretch of the dome-shape; and $\boldsymbol{\alpha}_2 \in \boldsymbol{\alpha}$ adjusts the vertical intercept (height) of f_q; $\boldsymbol{\alpha} > \mathbf{0}$.

According to Eq. (4.1), the decision variables to optimize are $\boldsymbol{\alpha}$, $\boldsymbol{\mu}$ and r, among which $r \in \mathbb{Z}_+$ and $\boldsymbol{\mu} \in \mathbb{Z}_+^2$ are integer variables in units of pixels of the input NHM. In

addition, r is a threshold in the indicator function $\mathbb{1}()$, making it difficult to derive its optimal value in close-form or by gradient descent.

Since this optimization is done individually for each local maximum, the total number of combinations of r and $\boldsymbol{\mu}$ to consider is in fact limited, especially considering that tree sizes are often not very large numbers in real-world urban environments. Thus, we enumerate through all combinations of $\boldsymbol{\mu}$ and r to obtain the optimal solutions. For parameter vector $\boldsymbol{\alpha}$, we have the following Theorems 6 and 7:

Theorem 6. *For a fixed combination of (r, $\boldsymbol{\mu}$), the solutions for $\boldsymbol{\alpha}$ can be evaluated in closed form:*

$$\boldsymbol{\alpha}_1 = \frac{(\sum_i \mathbb{1}_i)^{-1}(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)(\sum_i \mathbf{y}_i \mathbb{1}_i) - (\sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i)}{(\sum_i \mathbf{Z}_i^4 \mathbb{1}_i) - (\sum_i \mathbb{1}_i)^{-1} \cdot (\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)^2}$$

$$\boldsymbol{\alpha}_2 = [(\textstyle\sum_i \mathbf{y}_i \mathbb{1}_i) + \boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i)] / \sum_i \mathbb{1}_i$$

where $\mathbf{Z}_i = \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2$, $\mathbb{1}_i = \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r)$, $\mathbf{y}_i = \mathbf{y}(\mathbf{X}_i)$.

Proof. For a valid and fixed combination (r, $\boldsymbol{\mu}$), we can get rid of Constraints (4.2) and (4.3) since they must be satisfied. Thus, we are left with:

$$\begin{aligned}
&\|\big(\mathbf{y}(\mathbf{X}) - f_q(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\mu}, r)\big) \cdot \mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_2^2 / \|\mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_1 \\
=&\|\big(\mathbf{y}(\mathbf{X}) + (\boldsymbol{\alpha}_1 \cdot \mathbf{Z}^{\circ 2} + \boldsymbol{\alpha}_2) \cdot \mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_2^2 / \|\mathbb{1}(\mathbf{X}^T, \boldsymbol{\mu}, r)\|_1 \\
=&\textstyle\sum_i \big[\big(\boldsymbol{\alpha}_1^2 \mathbf{Z}_i^4 - 2\boldsymbol{\alpha}_1\boldsymbol{\alpha}_2\mathbf{Z}_i^2 + \boldsymbol{\alpha}_2^2 + 2\boldsymbol{\alpha}_1\mathbf{Z}_i\mathbf{y}_i - 2\boldsymbol{\alpha}_2\mathbf{y}_i + \mathbf{y}_i^2\big) \\
&\cdot \mathbb{1}_i\big] / \textstyle\sum_i \mathbb{1}_i
\end{aligned}$$

where $\mathbf{Z}^{\circ 2}$ denotes the Hadamard (element-wise) square of $\mathbf{Z}$.

Taking partial derivatives and setting them to 0s, we get:

$$\begin{cases}
\boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^4 \mathbb{1}_i) - \boldsymbol{\alpha}_2(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i) + \sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i = 0 \\
-\boldsymbol{\alpha}_1(\sum_i \mathbf{Z}_i^2 \mathbb{1}_i) + \boldsymbol{\alpha}_2(\sum_i \mathbb{1}_i) - \sum_i \mathbf{y}_i \mathbb{1}_i = 0
\end{cases} \tag{4.6}$$

Solving this linear system with two equations and two unknowns, we get the solutions stated in the theorem. □

Next we show that valid solutions are unique via Thm. 7.

Theorem 7. *The solutions of $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are unique when the mathematical surface is a valid approximation of a tree-shape, and the result of the negative quadratic function (Eq. (4.5)) is in the valid range $(0, h_{max}]$, where h_{max} is the maximum height of a tree in the spatial domain of interest.*

Proof. First, trees have concave shapes rather than convex or flat shapes in the normalized height models. This requires $\boldsymbol{\alpha}_1$ be positive in the negative quadratic function (Eq. (4.5)). If $\boldsymbol{\alpha}_1 \leq 0$, the combination or hypothesis (r, $\boldsymbol{\mu}$) should be directly rejected. Then, based on the derivatives, the only situation when the linear system does not have a single solution appears when $(\sum_i \mathbf{Z}_i^4) - (\sum_i \mathbb{1}_i)^{-1} \cdot (\sum_i \mathbf{Z}_i^2)^2$ (last term of $\boldsymbol{\alpha}_1$'s solution) is 0: (1) Possible outcome 1: infinite number of ($\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$) which satisfy the conditions in Eq. (4.6). However, in this case, the optimal solution for $\boldsymbol{\alpha}_1$ must be negative ($\boldsymbol{\alpha}_2$ must be positive) in order to achieve the minimum objective, according to the expansion of the objective function at the beginning of Thm. 6's proof (note that $\mathbf{y}(\mathbf{X}_i) \geq 0$, $\forall i$); (2) Possible outcome 2: $\boldsymbol{\alpha}_1$ becomes infinitely large. This is also invalid because $\boldsymbol{\alpha}_1 \to +\infty$ will make the result of Eq. (4.5) exceed h_{max} (also another type of non-tree shape). □

Optimization with Gaussian surfaces

Our TIMBER framework can be applied to general mathematical approximators. The negative quadratic function provides a concrete example for the polynomial family, and here we show TIMBER's use with a Gaussian approximator for the exponential family. We skip the proofs to reduce redundancy.

Since we use circles to approximate the 2D shapes of trees from a satellite view, the covariance matrix of the Gaussian function is diagonal and the diagonal elements are the same (i.e., same variance σ^2 for each direction). In addition, original Gaussian functions have inflection points (one dimension) or hyperplanes (multiple dimensions)

where the second order derivatives change signs (e.g., from a concave dome-shape to a convex bow-shape). Since trees have concave shapes, our approximator uses only the "concave" part, that is, the dome-surface towards the inside of the inflection boundary. In each direction of a Gaussian, the inflection points are located at $(\mu \pm \sigma)$. Thus, each element in $((\mathbf{X}_i)^T - \boldsymbol{\mu})$ is rescaled to the range of $[0, \pm\sigma]$. The Gaussian approximator is:

$$f_g\big((\mathbf{X}_i)^T, \boldsymbol{\alpha}, \boldsymbol{\mu}, r\big) = \boldsymbol{\alpha}_1 \cdot \frac{\exp(-\mathbf{d}_i^T(\sigma^2\mathbf{I})^{-1}\mathbf{d}_i)}{(\sqrt{(2\pi)^2|\sigma^2\mathbf{I}|})} + \boldsymbol{\alpha}_2 \tag{4.7}$$

where $\mathbf{d}_i = \frac{\sigma((\mathbf{X}_i)^T - \boldsymbol{\mu})}{r}$. The solutions to $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are given by (in the final solution σ is canceled out due to rescaling):

$$\boldsymbol{\alpha}_1 = \frac{\sum_i \mathbf{e}_i\mathbf{y}_i\mathbb{1}_i - (\sum_i \mathbb{1}_i)^{-1}(\sum_i \mathbf{y}_i\mathbb{1}_i)(\sum_i \mathbf{e}_i\mathbb{1}_i)}{(2\pi)^{-1}(\sum_i \mathbf{e}_i^2\mathbb{1}_i) - (2r\pi)^{-1}(\sum_i \mathbf{e}_i\mathbb{1}_i)^2}$$

$$\boldsymbol{\alpha}_2 = \big[2(\textstyle\sum_i \mathbf{y}_i\mathbb{1}_i) - \boldsymbol{\alpha}_1\pi^{-1}(\sum_i \mathbf{e}_i\mathbb{1}_i)\big] / \big[2(\textstyle\sum_i \mathbb{1}_i)\big]$$

where $\mathbf{e}_i = \exp(-\frac{\|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2^2}{2r^2})$.

Regularizations

The objective function (Eq. (4.1)) evaluates the mean square error of an approximator and selects the set of parameters that minimizes the mean error. While the mean is already a normalization of errors for different sizes (r) of an approximator, the objective function is still biased towards smaller sizes. Since the parameters $\boldsymbol{\alpha}$ are optimized to adjust the shape of the approximator to minimize the errors, it is easier to reduce the mean errors or even eliminate them with smaller number of $\mathbf{y}(\mathbf{X}_i)$. As an analogy, in linear regression, it is easier to fit a perfect hyperplane to a smaller number of points. We develop two regularizers to reduce the bias.

Vertical interval regularization: The bias towards smaller sizes mainly causes problems for large trees, which may have small "bumps" on top of their canopy, and

this may lead to underestimation of the tree size r. One characteristic of such "bumps" is that they also tend to have a very small range of height values in the vertical direction. Denote v_0 as $\inf\{\mathbf{y}(\mathbf{X}_i) \,|\, \forall i : \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r) = 1\}$, and v_1 as $\sup\{\mathbf{y}(\mathbf{X}_i) \,|\, \forall i : \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r) = 1\}$. The vertical interval is then $(v_1 - v_0)$. Since squared errors (L2 norm) are used in the objective function, here we use the squared vertical interval $(v_1 - v_0)^2$ to regularize the error. If we denote f_{obj} as the objective function, the regularized form is: $f_{obj}/(v_1 - v_0)^2$.

Minimum analysis window regularization: Vertical interval regularization cannot penalize the case when a small "bump" on a large tree can be perfectly fit by an approximator, because the zero error will be invariant to the ratio-based regularizer. Thus, we use a minimum analysis window size to further penalize this scenario. Denote w_{min} as the minimum size. If the local circular window formed by $\{(\mathbf{y}(\mathbf{X}_i), \mathbf{X}_i) \,|\, \forall i : \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r) = 1\}$ has a radius r smaller than w_{min}, it will be resampled to a higher resolution with size w_{min} (i.e., from a circular region inside a $(2r + 1) \times (2r + 1)$ window to a circular region inside a $(2w_{min} + 1) \times (2w_{min} + 1)$ window) using nearest-neighbor. Practically, we use the median of the input range of radii as w_{min}. The nearest-neighbor resampling method is used to reduce the chance of a small "bump" being perfectly fit by an approximator.

The two regularizers do not affect the closed form solutions for parameter $\boldsymbol{\alpha}$ in the approximators.

4.3.2 Phase 2: Deep Learning based Urban Tree Filter

The first phase finds tree-like structures using the approximators, but it can potentially include false detections of non-tree structures. In Phase 2, we remove the non-tree structures using a deep learning filter. One issue for machine learning in the tree detection problem is the unavailability of ground truth training data. Thus, rather than using a

deep learning model to detect the trees directly, we only use it as a filter, which is not trained on actual ground truth data but instead on a combination of Phase 1 detections, NHMs, and available city infrastructure data in a subset of study areas.

Many cities routinely collect and update digital information about building footprints, roads and other infrastructure such as street lights and utility towers. Such data can help determine if a detection in Phase 1 is more likely to be a tree or non-tree.

CNN-based Urban Tree Filter: The deep learning framework we use for this phase is a Convolutional Neural Network (CNN) [125]. Our CNN architecture takes input image patches of size $32 \times 32 \times 1$, and has two convolutional layers (kernel size 5×5), two max-pooling layers with strides of 2, and two fully connected layers at the end. Its training data includes a set of image patches and their labels. To construct the tree filter, we first extract a local image patch from the NHM for each detection $(\boldsymbol{\mu}^*, r^*)$ in the areas where city infrastructure data is available. For our tree filtering purpose, we are only interested in a binary labeling: [1: tree, 0: non-tree]. Using the city infrastructure data, we label an image patch as a "non-tree" if the center $\boldsymbol{\mu}^*$ of its corresponding detection is within the polygons of non-tree infrastructures (e.g., buildings, roads). The training dataset we use here is not perfect ground truth data, and it may contain some noise such as incorrect labels. Thus, the goal of this trained CNN-filter is not to detect trees, but to help remove non-tree objects when city infrastructure data is not available or not complete.

4.3.3 TIMBER Acceleration

The CNN in Phase 2 is in general very efficient during prediction. For Phase 1, we propose a Core Object REduction (CORE) algorithm for acceleration.

TIMBER_Base: A direct observation is that, for each combination of size and location $(r, \boldsymbol{\mu})$ at a local maximum, we only need to check the elements (e.g., $\mathbf{X}_i$, $\mathbf{y}_i$)

against the indicator function $\mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r)$ within the local square neighborhood of size $(2r+1, 2r+1)$ centered at $\boldsymbol{\mu}$. Further, the optimization processes at different local maxima are independent so they can be parallelized on multiple CPU cores.

TIMBER_CORE: While $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ can be evaluated in closed form, their computations can be expensive. For simplicity, here we use the solutions for the negative quadratic approximator (Thm. 6) as an example. The strategy is the same for the Guassian approximator.

Table 4.1: Summation units in $\boldsymbol{\alpha}$ solutions

	$\mathbf{X}$, $\mathbb{1}$	$\mathbf{y}$, $\mathbb{1}$	$\mathbf{X}$, $\mathbf{y}$, $\mathbb{1}$	$\mathbb{1}$
$\boldsymbol{\alpha}_1$	$\sum_i \mathbf{Z}_i^2 \mathbb{1}_i$, $\sum_i \mathbf{Z}_i^4 \mathbb{1}_i$	$\sum_i \mathbf{y}_i \mathbb{1}_i$	$\sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i$	$\sum_i \mathbb{1}_i$
$\boldsymbol{\alpha}_2$	$\sum_i \mathbf{Z}_i^2 \mathbb{1}_i$	$\sum_i \mathbf{y}_i \mathbb{1}_i$	-	$\sum_i \mathbb{1}_i$

*Notation: $\mathbf{Z}_i = \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2$, $\mathbb{1}_i = \mathbb{1}((\mathbf{X}_i)^T, \boldsymbol{\mu}, r)$, and $\mathbf{y}_i = \mathbf{y}(\mathbf{X}_i)$.

In Thm. 6, there are many summation operations needed to compute elements in $\mathbf{X}$, $\mathbf{y}$ and $\mathbb{1}$ as well as their cross products. These summations introduce most of the computations. Here we consider each summation as a summation unit. Table 4.1 lists all summation units without duplicates. The row names identify the parameter that the summation units belong to, and the column names show where the participating elements in the summations are from. The idea of the CORE algorithm is to identify the core objects in the solutions that can be shared across multiple optimization processes to reduce computation.

First, examining the values of $\mathbf{Z}_i = \|(\mathbf{X}_i)^T - \boldsymbol{\mu}\|_2$, we can find that although the values of $\mathbf{X}_i$ and $\boldsymbol{\mu}$ can all differ, their differences remain the same for a fixed radius

r across all locations. For example, consider the local windows of radius r (in pixels) centered at all $\boldsymbol{\mu}$. If we compute the $\mathbf{Z}_i$ values for the pixels in all these local windows, all resulting matrices will be identical despite their different $\mathbf{X}_i$ and $\boldsymbol{\mu}$ values. Furthermore, all these local windows of a fixed size r will share the same indicator function values as well. Combining these two observations, we can conclude that the results of all summation units of $\sum_i \mathbf{Z}_i^2 \mathbb{1}_i$, $\sum_i \mathbf{Z}_i^4 \mathbb{1}_i$ and $\sum_i \mathbb{1}_i$ are the same for all optimizations with the same size r.

For summation units $\sum_i \mathbf{Z}_i^2 \mathbf{y}_i \mathbb{1}_i$ and $\sum_i \mathbf{y}_i \mathbb{1}_i$, their values are different at each location $\boldsymbol{\mu}$ because $\mathbf{y}_i$ values can differ across locations and they are not neutralized by $\boldsymbol{\mu}$ as in $\mathbf{Z}_i$. Nevertheless, here we can see that all these summations still involve the values of the indicator function $\mathbb{1}$ and $\mathbf{Z}_i$. Since for each size r, the values of $\mathbb{1}$ and $\mathbf{Z}_i$ remain the same in all the local windows, we can keep their values in two matrices of size $(2r+1, 2r+1)$, that is, the size of the minimum bounding square of a local window with a radius r. These matrices can then be shared across all optimizations of the same size r.

All these core objects for all candidate sizes $\{r\}$ only need to be computed once at the beginning of TIMBER; they then become inputs to the computations of $\boldsymbol{\alpha}$.

4.3.4 Computational Complexity Analysis

Here we evaluate the time complexity of the first phase of TIMBER as it is the computational bottleneck. Denote the number of local maxima and tree sizes as N and M, respectively, the maximum tree size (in pixels) as r_{max}, the window size for the flexible tree center search as w, and the number of CPU cores as C. The CORE algorithm reduces the necessary number of summations (Table 4.1) to a constant portion ρ of all summations. While it does not change the asymptotic complexity of the optimization process, which is $O(C^{-1}NMw^2r_{max}^2)$, the constant scaling could still result in a

significant amount of savings for a long execution time.

4.4 Validation

We validated the solution quality of TIMBER through a case study, and confirmed the computational savings achieved by the CORE algorithm using controlled experiments.

4.4.1 Case Study

We conducted the case study in Minneapolis, US, which is a well populated and urbanized region with a well-maintained green infrastructure (e.g., trees) across its district zones.

The total area of the zones in our case study was about 6,500 acres, one third of which was used to generate the training data of the second phase of TIMBER (i.e., CNN-based urban tree filter). Transfered learning was used to facilitate the convergence. Overall, we implemented 6 candidate algorithms for tree detection in this comparison:

(1) *TIMBER$_Q$:* TIMBER framework with the approximator of negative quadratic functions. (2) *TIMBER$_G$:* TIMBER with the Gaussian approximator. (3) *YOLO:* You-Only-Look-Once (YOLO) [116] is a state-of-the-art convolutional neural network for object detection. Since we did not have actual ground truth data, YOLO was trained with the same data used by our CNN classifier in Phase 2, which was the best that we could do. (4) *Watershed-SEG:* Watershed segmentation is the most used framework for tree detections in remote sensing communities [117, 119, 120]. (5) *Spatial-SEG:* Mean-shift segmentation [126, 122] is a commonly used method for segmenting spatial datasets which considers both spatial and spectral similarities. (6) *GMM-EM:* The Gaussian-Mixture-Model (GMM) clustering algorithm with Expectation-Maximization (EM) [114] has also been used to estimate tree canopy sizes. We used height and locations as the features in GMM, and initialized its EM optimization with the local

maxima in the NHM to facilitate the convergence. Since we did not know the number of clusters (trees), we used the number of local maxima as an estimate.

Result postprocessing: We used digital city infrastructure data (e.g., buildings, towers, roads, etc.) to filter out the non-tree detections in Watershed-SEG, Spatial-SEG and GMM-EM (otherwise much lower accuracy). This is not doable for regions without such data. In addition, the irregular tree shapes detected by the three methods were approximated to circles.

Since individual tree locations and canopy sizes are rarely collected or publicly available, we selected four areas in the test region (not seen in training), and went through a time-consuming manual inspection to generate the ground truth data for testing. We also involved spatial data experts to help improve the data collection using functionalities of spatial software (e.g., 3D profiling, visualization enhancement).

Fig. 4.2 visualizes an example of the data, ground truth, and representative results from four candidate methods. As we can see, TIMBER better captured the tree locations and canopy sizes in the ground truth. Since watershed segmentation is a rigid segmentation based on geometric properties, it got stuck in the local fluctuations on large tree tops, splitting a single tree into many smaller pieces. YOLO's main limitation is on detecting small objects appearing in groups, so it experienced difficulties in finding and separating out the trees.

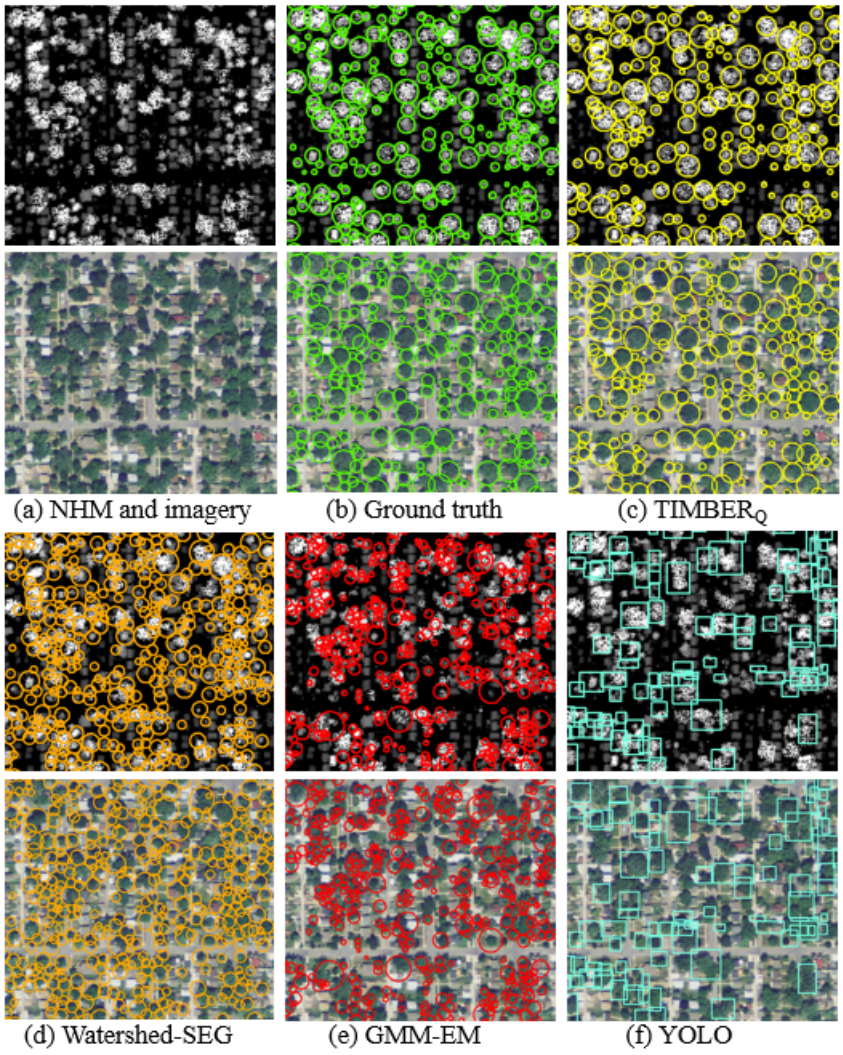


Figure 4.2: Comparison of detections. The imagery was collected one year after the NHM, so several trees (removed) may only show up in the NHM. The imagery is only used for visual assistance. (best in color)

The four test areas had different landscapes (e.g., mixtures of trees and small or large buildings). The number of trees in test Areas 1 to 4 was 1270, 1393, 972 and 1193, respectively. We evaluated the precision, recall and F1-scores for the 6 methods

over the four test areas. The results are presented in Fig. 4.3 (a)-(c), and detailed in Tables 4.2, 4.3 and 4.4. The statistics are consistent with our visual comparison in Fig. 4.2. TIMBER$_Q$ achieved the highest precision (90-95%), recall (80-85%) and F1-scores (85-90%) in all four test areas. While TIMBER$_G$ also performed consistently better than related work, there was about a 10% gap with TIMBER$_Q$. This is an interesting result since it suggests that negative quadratic function is a better approximator of tree shapes compared to Gaussian. Among approaches from related work, YOLO achieved the highest precision (50-60%) but had low recall (∼25%), leading to low F1 scores, while Watershed-SEG had good recall (∼65%) but suffered from low precision (40-45%). This could be due to its tendency to split individual large trees into smaller pieces. Neither Spatial-SEG nor GMM-EM performed well in the test areas.

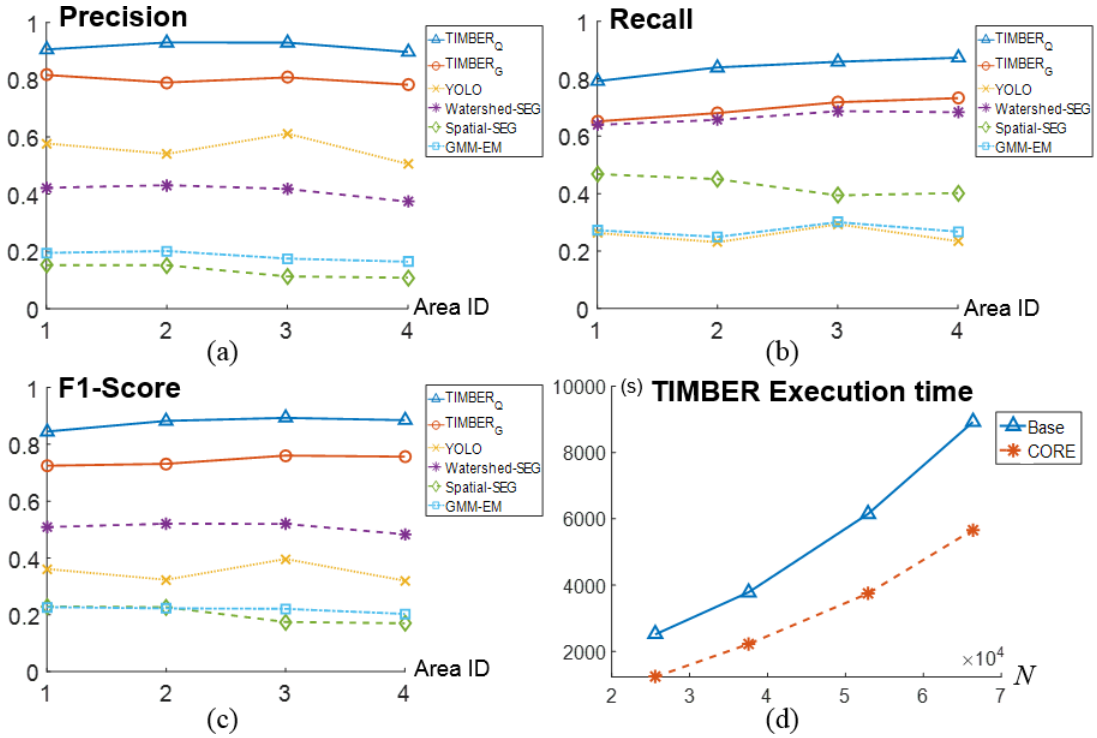


Figure 4.3: Solution quality statistics and execution time.

Table 4.2: Precision of detections in test areas

Methods	**Area 1**	**Area 2**	**Area 3**	**Area 4**
TIMBER$_Q$	91.5%	94.4%	94.6%	91.8%
TIMBER$_G$	82.5%	80.5%	82.1%	81.2%
YOLO	57.7%	54.5%	62.3%	51.8%
Watershed-SEG	43.7%	43.6%	42.3%	39.2%
Spatial-SEG	16.1%	15.7%	11.6%	11.9%
GMM-EM	20.6%	21.2%	18.0%	17.8%

Table 4.3: Recall of detections in test areas

Methods	**Area 1**	**Area 2**	**Area 3**	**Area 4**
TIMBER$_Q$	77.8%	82.1%	86.2%	85.2%
TIMBER$_G$	63.8%	66.4%	71.8%	71.5%
YOLO	25.7%	22.1%	29.4%	22.5%
Watershed-SEG	63.9%	63.8%	68.4%	67.5%
Spatial-SEG	48.0%	45.1%	39.9%	41.3%
GMM-EM	27.7%	24.9%	30.5%	27.2%

4.4.2 Computational Performance

In the complexity analysis of the optimization process (Sec. 4.3.4), most of the parameters are related to the maximum tree size r_{max}. In real world applications, especially in urban environments, r_{max} is often fairly limited (e.g., 15, 20 meters). Thus, compared to the total number of local maxima N, r_{max} and other parameters are relatively stable (similar to constants). Thus, our analysis focused on the effect of N, which was varied by changing the size of the study area.

We evaluated the proposed algorithms on a 24-core computing node in a Linux

Table 4.4: F1 scores of detections in test areas

Methods	**Area 1**	**Area 2**	**Area 3**	**Area 4**
TIMBER$_Q$	84.1%	87.8%	90.2%	88.4%
TIMBER$_G$	71.9%	72.8%	76.6%	76.1%
YOLO	35.5%	31.5%	40.0%	31.4%
Watershed-SEG	51.9%	51.8%	52.3%	49.6%
Spatial-SEG	24.1%	23.2%	17.9%	18.5%
GMM-EM	23.6%	22.9%	22.7%	21.5%

environment. Since the results were very similar for TIMBER$_Q$ and TIMBER$_G$, here we present those of the former for illustration purposes. Fig. 4.3(d) shows the execution time for the baseline and CORE algorithms. As discussed in Sec. 4.3.4, CORE does not change the asymptotic complexity but reduces the number of summation units (Table 4.1) to a constant proportion. Through Fig. 4.3(d) we can see that the speedup is about 1.5X to 2X in the experiments. In the largest area studied (i.e., a 3.5×2.5km^2 region corresponding to the maximum N), the CORE algorithm saved more than 20 hours of CPU time (about one hour of wall-time).

4.5 Conclusions and Future Work

We proposed a two-phase TIMBER framework to generate individual tree inventories from remote sensing datasets as well as a CORE algorithm to improve computational efficiency. Through experiments, we showed that TIMBER can significantly improve accuracy compared to related work and the CORE algorithm can speed up the computation. In future work, we aim to generate a benchmark tree inventory to facilitate future research on tree species classification at a real-world large scale. In addition, we aim to design new acceleration methods to further reduce the computational time.

Chapter 5

Conclusions and Future Research Directions

5.1 Key Results

One size AI does not fit all. Spatial data and its vast array of applications pose major challenges to traditional AI techniques, including high cost of spurious results, spatial interdependency and data gaps. This thesis addressed some of these challenges for three types of GeoAI techniques, i.e., learning (e.g., unsupervised clustering), planning (e.g., spatial optimization) and perception (e.g., geospatial object mapping), in the context of applications for smart cities and communities. First, the thesis proposed a significant DBSCAN for statistically robust clustering, introducing a modeling of statistical significance in DBSCAN as well as a dual-convergence algorithm to accelerate the computation. Second, the thesis proposed a fragmentation-free spatial allocation technique to model spatial interdependency during optimization. It introduced a new formulation with spatial decision variables to explicitly model spatial contiguity and regularity constraints (i.e., each contiguous patch formed by choices on nearby decision variables

must satisfy both a minimum area and a regular-shape constraints), and presented a fast hierarchical-fragmentation-elimination algorithm to efficiently solve the problem in a heuristic manner. Finally, the thesis proposed a domain-knowledge assisted deep learning method called TIMBER to address the challenge of limited training data in the mapping of trees in complex urban environments. Extensive experiments were conducted to validate these methods through both comparative and sensitivity analyses. The results confirmed that the proposed approaches can greatly improve solution quality in both synthetic and real-world data, and the proposed algorithms can greatly improve the computational performance.

Table 5.1 summarizes these key results. It also outlines the future research directions I plan to take in the both short-term (colored in blue) and the long-term (colored in green).

5.2 Short-Term Future Research Directions

In the short-term, I plan to further improve GeoAI learning (e.g., unsupervised clustering) by bridging remaining research gaps caused by the three challenges listed in Table 5.1.

A general framework for statistically-robust unsupervised spatial clustering: In this thesis, Ch. 2 proposed a significance modeling and acceleration technique specifically designed for incorporating statistical rigor into DBSCAN. While DBSCAN is one of the most popular unsupervised clustering techniques, there also exist many other widely used clustering methods such as H-DBSCAN [88], OPTICS [87], CHAMELEON [127], spectral clustering [128], Gaussian mixture models [114] (e.g., k-means, EM), etc. Rather than designing a specific method for each technique, I aim to study a more general framework which can be easily customized and applied to different unsupervised clustering methods. To start with, some ideas from significant DBSCAN (Ch.

Table 5.1: Taxonomy of thesis contributions and future research directions for short-term (ST, colored in blue) and long-term (LT, colored in green).

	GeoAI Techniques		
Challenges	**Learning (e.g., unsupervised clustering)**	**Planning (e.g., spatial optimization)**	**Perception (e.g., object mapping)**
High cost of spurious results	• Significant DBSCAN (Ch. 2) • A general framework for statistically robust spatial clustering (ST)	• Uncertainty-explicit (e.g., data accuracy, extreme events) fragmentation-free spatial allocation (LT)	• Uncertainty-explicit (e.g., bounds) mapping and segmentation (LT)
Spatial interdependency and variability (e.g., non-stationarity, gerrymandering)	• Interdependency-aware hypotheses for significant DBSCAN (ST) • Variability-explicit framework for deep learning (LT)	• Fragmentation-free spatial allocation (Ch. 3) • Variability-aware fragmentation-free spatial allocation (LT)	• Context-aware object selection/ filtering (Ch. 4)
Data and domain knowledge gap	• Multi-source (e.g., remote sensing imagery, survey, mobile) based integrative pattern recognition (ST)	• Physics-assisted optimization for transportation management (LT)	• Domain-knowledge assisted deep learning for object mapping (Ch. 4)

2) can be potentially transformative. For example, the dual-convergence algorithm designed to accelerate significance testing for both significant and spurious patterns can be generalized into a higher-level computational framework to work with other clustering techniques.

Interdependency-aware hypotheses for significant DBSCAN: Currently, the hypothesis used for significant DBSCAN assumes independence among data points. However, this may not be ideal for many applications where interaction or correlation exists between nearby data points. In agriculture, for example, many events (e.g., disease, pest, water stress) are spatially-autocorrelated as a result of the underlying

physical processes (e.g., the spread of a disease). As a result, hypotheses that ignore the interdependency among data points may not correctly reflect the true phenomenon and lead to inaccurate results. In the next phase of my research, I plan to first investigate the effect of spatial interdependency on significance testing results and then collaborate with domain scientists to develop interdependency-aware hypotheses to improve solution quality.

Spatial-variability-explicit framework for deep learning: As discussed in Sec. 1.3, spatial data exhibit both interdependency and variability. Recent developments in deep learning have recognized and addressed the issue of interdependency via convolutional neural networks, in which locally-connected layers are used to replace fully-connected layers in traditional architectures to explicitly model spatial interdependency (i.e., nearby samples are more similar than distant ones). However, spatial variability has not yet been sufficiently studied, and this significantly limits the value of these models in large-scale applications. For example, in agriculture, optimal parameters for yield prediction models often differ across geographic regions due to unknown or unobserved differences in the complex environmental or social context. While spatial ensemble methods (e.g., [32]) have been studied to partition data and learn simpler local models (e.g., decision tree), they may not be well-suited for deep learning techniques which require significantly more training data to learn millions of parameters. I will investigate new frameworks to balance spatial-variability-awareness and the need for large training data.

Multi-source based integrative pattern recognition: The success of spatial pattern recognition techniques not only depends on the quality of the approach but also the quality of the data. In many scenarios, the dataset available for use in one technique is not representative of the true phenomenon as discussed earlier in Sec. 1.3. For example, in a disease outbreak, smartphone trajectories may not form a reliable

dataset on its own to reveal changes in mobility patterns due to selection bias. I plan to investigate novel pattern recognition techniques that can integrate a variety of complimentary data sources (e.g., remote sensing imagery, surveys) to better model and learn patterns in real-world scenarios. In the previous example of disease outbreak, traffic volume extracted from high-frequency remote sensing imagery may provide additional insights to learn the changes in mobility patterns beyond smartphone trajectories.

5.3 Long-Term Future Research Directions

In the long-term, I plan to further address the three challenges (Table 5.1) in the context of planning (e.g., spatial optimization in allocation and routing) and perception (e.g., geospatial object mapping).

Uncertainty-explicit fragmentation-free spatial allocation: Uncertainty needs to be explicitly considered to improve the stability of allocation plans from fragmentation-free spatial allocation and reduce the risk of spurious results (i.e., a better solution achieved through optimization turning out to be worse than the previous plan after being implemented in real-world scenarios). The uncertainty may come from various sources, including errors in the data (e.g., simulation errors from agricultural models), dynamics in markets (e.g., price changes of crops) and extreme events (e.g., unexpected floods or droughts). These factors need to be explicitly considered and expressed in the outputs to better inform real-world stakeholders.

Uncertainty-explicit mapping and segmentation for remote sensing data: Uncertainty also needs to be explicitly modeled in geospatial mapping and segmentation tasks to avoid spurious results that can lead to high-cost failures in many applications. As illustrated in Sec. 1.3, errors in geo-imagery based mapping (e.g., ice, open water) may cause ships to get stuck in ice floes during long-distance travels. To overcome this challenge, uncertainty in such sensitive classification tasks needs to be explicitly

modeled (e.g., Dempster-Shafer theory) and clearly presented to decision makers so they are aware of the real risks. This will also require high interpretability of the techniques in order to understand and model the propagation of uncertainty and errors.

Physics assisted optimization for transportation management: Data gaps are a major challenge not only in GeoAI learning, but also in many optimization based planning tasks. Decision optimization (e.g., signal control, routing) in transportation is an important area that faces this challenge. Transportation accounts for about one-third of total energy consumption in the US and is a major source of greenhouse gas emissions. There exists a great opportunity to save energy and reduce emissions by promoting energy-efficient and low-emission decisions in transportation. The opportunity is especially exciting with new data sources such as on-board-diagnostic data, which record hundreds of engine measurements (e.g., fuel consumption, emission, speed, acceleration) along trajectories. However, trajectories in such data may only cover a subset of road segments in large transportation networks, and this data incompleteness significantly limits the use of optimization algorithms for transportation such as routing or signal control. To bridge this data gap, I plan to investigate a physics assisted optimization framework which combines engine physics models with existing on-board-diagnostic data on a subset of transportation network to estimate the energy consumption as well as emission on road segments with limited data availability.

Spatial-variability-aware optimization: As introduced in Sec. 1.3, a fundamental property of spatial data is that data distribution is non-stationary across geographic regions. This spatial variability poses a big challenge for spatial optimization. In fragmentation-free spatial allocation, for example, the spatial constraints (e.g., minimum area constraint, shape constraint) used to enforce spatial contiguity and regularity of the output land allocation design are assumed to be the same across the entire study

area. This assumption may not be appropriate for large-scale applications where different sub-regions of the study area require different spatial constraints due to physical (e.g., local topographical characteristics) or social (e.g., farmers' preferences) reasons. Thus, the next generation of fragmentation-free spatial allocation methods will need to incorporate spatial variability awareness into the optimization formulations as well as algorithms to explicitly model such non-stationarity and solve applications at large-scales.

References

[1] Whats driving the connected car? https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/whats-driving-the-connected-car, 2014.

[2] Continuous evolution. https://earthdata.nasa.gov/esds/continuous-evolution, 2020.

[3] Yiqun Xie, Jayant Gupta, Yan Li, and Shashi Shekhar. Transforming smart cities with spatial computing. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–9. IEEE, 2018.

[4] Yiqun Xie, Shashi Shekhar, Richard Feiock, and Joseph Knight. Revolutionizing tree management via intelligent spatial techniques. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 71–74, 2019.

[5] S&cc-irg track 1: Connecting the smart-city paradigm with a sustainable urban infrastructure systems framework to advance equity in communities. `https://www.nsf.gov/awardsearch/showAward?AWD_ID=1737633&HistoricalAwards=false`, 2017.

[6] Reem Y Ali, Venkata MV Gunturi, Andrew J Kotz, Shashi Shekhar, and William F Northrop. Discovering non-compliant window co-occurrence patterns:

A summary of results. In *International Symposium on Spatial and Temporal Databases*, pages 391–410. Springer, 2015.

[7] Yan Li, Shashi Shekhar, Pengyue Wang, and William Northrop. Physics-guided energy-efficient path selection: a summary of results. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108, 2018.

[8] Yan Li, Pratik Kotwal, Pengyue Wang, Yiqun Xie, Shashi Shekhar, and William Northrop. Physics-guided energy-efficient path selection using on-board diagnostics data. *ACM Transactions on Data Science*, under review.

[9] Yiqun Xie and Shashi Shekhar. Ff-sa: Fragmentation-free spatial allocation. In *International Symposium on Spatial and Temporal Databases*, pages 319–338. Springer, 2017.

[10] Yiqun Xie, KwangSoo Yang, Shashi Shekhar, Brent Dalzell, and David Mulla. Geodesign optimization (GOP) towards improving agricultural watershed sustainability. In *Thirty-First AAAI Conference on Artificial Intelligence, Workshop on AI and OR for Social Good*, AAAI-17, pages 57–63, 2017.

[11] Yiqun Xie, Bryan C Runck, Shashi Shekhar, Len Kne, David Mulla, Nicolas Jordan, and Peter Wiringa. Collaborative geodesign and spatial optimization for fragmentation-free land allocation. *ISPRS International Journal of Geo-Information*, 6(7):226, 2017.

[12] Geoglam. http://earthobservations.org/geoglam.php, 2020.

[13] Naoki Abe, Yiqun Xie, Shashi Shekhar, Chid Apte, Vipin Kumar, Mitch Tuinstra, and Ranga Raju Vatsavai. Data science for food, energy and water: A workshop report. *ACM SIGKDD Explorations Newsletter*, 18(2):1–4, 2017.

[14] Yiqun Xie et al. Transdisciplinary foundations of geospatial data science. *ISPRS International Journal of Geo-Information*, 6(12):395, 2017.

[15] Martin Kulldorff. A spatial scan statistic. *Comm. in Statistics-Theory and methods*, 26(6):1481–1496, 1997.

[16] Yiqun Xie et al. A nondeterministic normalization based scan statistic (nn-scan) towards robust hotspot detection: a summary of results. In *SIAM Intl. Conf. on Data Mining (SDM'19)*, 2019.

[17] Yiqun Xie and Shashi Shekhar. A unified framework for robust and efficient hotspot detection in smart cities. *ACM Transactions on Data Science (TDS)*, in press, 2020.

[18] Daniel B Neill and Andrew W Moore. Rapid detection of significant spatial clusters. In *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 256–265, 2004.

[19] Emre Eftelioglu et al. Ring-shaped hotspot detection: a summary of results. In *Data Mining (ICDM), 2014 IEEE Intl. Conf. on*, pages 815–820. IEEE, 2014.

[20] Yiqun Xie and Shashi Shekhar. Significant dbscan towards statistically robust clustering. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, pages 31–40, 2019.

[21] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and data engineering*, 16(12):1472–1485, 2004.

[22] Jin Soung Yoo and Shashi Shekhar. A joinless approach for mining spatial colocation patterns. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1323–1337, 2006.

[23] Sajib Barua and Jörg Sander. Mining statistically significant co-location and segregation patterns. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1185–1199, 2013.

[24] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):1977–1992, 2011.

[25] Xun Zhou, Shashi Shekhar, Pradeep Mohan, Stefan Liess, and Peter K Snyder. Discovering interesting sub-paths in spatiotemporal datasets: A summary of results. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 44–53, 2011.

[26] Yiqun Xie, Xun Zhou, and Shashi Shekhar. Discovering interesting subpaths with statistical significance from spatiotemporal datasets. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(1):1–24, 2020.

[27] Xun Zhou, Shashi Shekhar, and Dev Oliver. Discovering persistent change windows in spatiotemporal datasets: A summary of results. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 37–46, 2013.

[28] Xun Zhou, Shashi Shekhar, and Reem Y Ali. Spatiotemporal change footprint pattern discovery: an inter-disciplinary survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(1):1–23, 2014.

[29] Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 371–376, 2001.

[30] Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. A unified approach to detecting spatial outliers. *GeoInformatica*, 7(2):139–166, 2003.

[31] Zhe Jiang, Shashi Shekhar, Xun Zhou, Joseph Knight, and Jennifer Corcoran. Focal-test-based spatial decision tree learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(6):1547–1559, 2014.

[32] Zhe Jiang, Yan Li, Shashi Shekhar, Lian Rampi, and Joseph Knight. Spatial ensemble learning for heterogeneous geographic data with class ambiguity: A summary of results. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2017.

[33] Sanjay Chawla, Shashi Shekhar, Weili Wu, and Uygar Ozesmi. Modeling spatial dependencies for mining geospatial data. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. SIAM, 2001.

[34] Shashi Shekhar, Paul R Schrater, Ranga Raju Vatsavai, Weili Wu, and Sanjay Chawla. Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia*, 4(2):174–188, 2002.

[35] Mete Celik, Baris M Kazar, Shashi Shekhar, and Daniel Boley. Parameter estimation for the spatial autoregression model: A rigorous approach£. In *Proceedings of the 2nd NASA Data Mining Workshop: Issues and Applications in Earth Science with the 38th Symposium on the Interface of Computing Science, Statistics and Applications*, 2006.

[36] Qingsong Lu, Betsy George, and Shashi Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. In *International symposium on spatial and temporal databases*, pages 291–307. Springer, 2005.

[37] KwangSoo Yang, Apurv Hirsh Shekhar, Dev Oliver, and Shashi Shekhar. Capacity-constrained network-voronoi diagram. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):2919–2932, 2015.

[38] Betsy George and Shashi Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. In *Journal on Data Semantics XI*, pages 191–212. Springer, 2008.

[39] Junming Liu, Qiao Li, Meng Qu, Weiwei Chen, Jingyuan Yang, Hui Xiong, Hao Zhong, and Yanjie Fu. Station site optimization in bike sharing systems. In *2015 IEEE International Conference on Data Mining*, pages 883–888. IEEE, 2015.

[40] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014, 2016.

[41] Yiqun Xie, Han Bao, Shashi Shekhar, and Joseph Knight. A timber framework for mining urban tree inventories using remote sensing datasets. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1344–1349. IEEE, 2018.

[42] Yiqun Xie, Rahul Bhojwani, Shashi Shekhar, and Joseph Knight. An unsupervised augmentation framework for deep learning based geospatial object detection: a summary of results. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 349–358, 2018.

[43] Yiqun Xie, Jiannan Cai, Rahul Bhojwani, Shashi Shekhar, and Joseph Knight. A locally-constrained yolo framework for detecting small and densely-distributed building footprints. *International Journal of Geographical Information Science*, pages 1–25, 2019.

[44] Chenyi Zhuang, Nicholas Jing Yuan, Ruihua Song, Xing Xie, and Qiang Ma. Understanding people lifestyles: Construction of urban movement knowledge graph from gps trajectory. In *IJCAI*, pages 3616–3623, 2017.

[45] Bo Yan, Krzysztof Janowicz, Gengchen Mai, and Rui Zhu. A spatially explicit reinforcement learning model for geographic knowledge graph summarization. *Transactions in GIS*, 23(3):620–640, 2019.

[46] Oracle spatial. https://www.oracle.com/database/technologies/spatialandgraph.html, 2020.

[47] Shaowen Wang and Michael F Goodchild. *CyberGIS for geospatial discovery and innovation*. Springer, 2019.

[48] Sushil K Prasad et al. Parallel processing over spatial-temporal datasets from geo, bio, climate and social science communities: A research roadmap. In *Big Data, 2017 IEEE Intl. Congress on*, pages 232–250.

[49] Spatialhadoop. http://spatialhadoop.cs.umn.edu/, 2020.

[50] Ahmed Eldawy and Mohamed F. Mokbel. SpatialHadoop: A MapReduce Framework for Spatial Data. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 1352–1363, 2015.

[51] Geospark. http://geospark.datasyslab.org/, 2020.

[52] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. Spatial data management in apache spark: The geospark perspective and beyond. *Geoinformatica*, 23(1):37–78, 2019.

[53] Here hd live map. https://www.here.com/products/automotive/hd-maps, 2020.

[54] Farmbeats: Ai, edge & iot for agriculture. https://www.microsoft.com/en-us/research/project/farmbeats-iot-agriculture/, 2020.

[55] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 515–529, 2017.

[56] U. Irfan. California's largest utility just declared bankruptcy. Hello, climate change. https://www.vox.com/2019/1/14/18182162/pg-e-camp-fire-bankruptcy, 2018.

[57] California schools close as smoke from camp fire fills the skies. https://www.nbcnews.com/news/us-news/california-schools-close-smoke-camp-fire-fills-skies-n936961, 2018.

[58] Minnesota cities struggle to stay ahead of emerald ash borer's rapid spread, 2019.

[59] Daniel B Neill. Expectation-based scan statistics for monitoring spatial time series data. *International Journal of Forecasting*, 25(3):498–517, 2009.

[60] Daniel B Neill and Gregory F Cooper. A multivariate bayesian scan statistic for early event detection and characterization. *Machine learning*, 79(3):261–282, 2010.

[61] Daniel B Neill, Andrew W Moore, and Gregory F Cooper. A bayesian spatial scan statistic. In *Advances in neural information processing systems*, pages 1003–1010, 2006.

[62] Xia Jiang and Gregory F Cooper. A bayesian spatio-temporal method for disease outbreak detection. *Journal of the American Medical Informatics Association*, 17(4):462–471, 2010.

[63] Minnesota buffer law, 2017.

[64] C S Slotterback, Bryan C. Runck, DG Pitt, Len Kne, and NR Jordan. Collaborative geodesign to advance multifunctional landscapes. *Landscape and Urban Planning*, in press:71–80, 2016.

[65] Voting rights were already a big 2020 issue. then came the gerrymandering ruling. https://www.nytimes.com/2019/06/28/us/politics/supreme-court-gerrymandering-democrats.html, 2019.

[66] Daniel B Neill and Andrew W Moore. A fast multi-resolution method for detection of significant spatial disease clusters. In *Advances in Neural Information Processing Systems (NIPS)*, pages 651–658, 2004.

[67] Emre Eftelioglu et al. Ring-shaped hotspot detection. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3367–3381, 2016.

[68] ULF Hjalmars, Martin Kulldorff, GÖRAN GUSTAFSSON, and Neville Nagarwalla. Childhood leukaemia in sweden: using gis and a spatial scan statistic for cluster detection. *Statistics in medicine*, 15(7-9):707–715, 1996.

[69] National Cancer Institute. https://surveillance.cancer.gov/satscan/, 2017.

[70] Safety culture and the zero deaths vision. `https://safety.fhwa.dot.gov/zerodeaths/`, 2020.

[71] Lei Shi and Vandana P Janeja. Anomalous window discovery through scan statistics for linear intersecting paths (sslip). In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–776. ACM, 2009.

[72] Vandana Pursnani Janeja and Vijayalakshmi Atluri. Ls 3: A linear semantic scan statistic technique for detecting anomalous windows. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 493–497. ACM, 2005.

[73] Dev Oliver, Shashi Shekhar, James M Kang, Renee Laubscher, Veronica Carlan, and Abdussalam Bannur. A k-main routes approach to spatial network activity summarization. *IEEE transactions on knowledge and data engineering*, 26(6):1464–1478, 2013.

[74] Xun Tang, Emre Eftelioglu, Dev Oliver, and Shashi Shekhar. Significant linear hotspot discovery. *IEEE Transactions on Big Data*, 3(2):140–153, 2017.

[75] John W Coulston and Kurt H Riitters. Geographic analysis of forest health indicators using spatial scan statistics. *Environmental management*, 31(6):764–773, 2003.

[76] Songlin Fei. Applying hotspot detection methods in forestry: a case study of chestnut oak regeneration. *International Journal of Forestry Research*, 2010, 2010.

[77] Julia Krolik, Gerald Evans, Paul Belanger, Allison Maier, Geoffrey Hall, Alan Joyce, Stephanie Guimont, Amanda Pelot, and Anna Majury. Microbial source tracking and spatial analysis of e. coli contaminated private well waters in southeastern ontario. *Journal of water and health*, 12(2):348–357, 2014.

[78] Julia Krolik, Allison Maier, Gerald Evans, Paul Belanger, Geoffrey Hall, and Alan Joyce. A spatial analysis of private well water escherichia coli contamination in southern ontario. *Geospatial Health*, 8(1):65–75, 2013.

[79] Shashi Shekhar, Steven Feiner, and Walid Aref. Spatial computing. *Communications of the ACM*, 59(1):72–81, 2015.

[80] SaTScan. https://www.satscan.org/, 2017.

[81] Emre Eftelioglu, Xun Tang, and Shashi Shekhar. Geographically robust hotspot detection: a summary of results. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 1447–1456. IEEE, 2015.

[82] Martin Kulldorff, Lan Huang, Linda Pickle, and Luiz Duczmal. An elliptic spatial scan statistic. *Statistics in medicine*, 25(22):3929–3943, 2006.

[83] Martin Kulldorff, Lan Huang, and Linda Pickle. An elliptic spatial scan statistic and its application to breast cancer mortality data in northeastern united states. *Journal of Urban Health*, 80:i130–i131, 2003.

[84] Emre Eftelioglu, Yan Li, Xun Tang, Shashi Shekhar, James M Kang, and Christopher Farah. Mining network hotspots with holes: A summary of results. In *International Conference on Geographic Information Science*, pages 51–67. Springer, 2016.

[85] Renato Assuncao, M Costa, A Tavares, and S Ferreira. Fast detection of arbitrarily shaped disease clusters. *Statistics in medicine*, 25(5):723–742, 2006.

[86] Martin Ester et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Intl. Conf. on Knowledge Discovery and Data Mining*, KDD'96, 1996.

[87] Mihael Ankerst et al. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[88] Ricardo Campello et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):5, 2015.

[89] Daniel B Neill. Detection of spatial and spatio-temporal clusters. In *Tech Rep CMU-CS-06-142, PhD thesis*. Carnegie Mellon University, 2006.

[90] HDBSCAN. https://hdbscan.readthedocs.io/en/latest/api.html, 2019.

[91] Daniel B Neill et al. Rapid detection of significant spatial clusters. In *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 256–265, 2004.

[92] Minneapolis sets winter parking restrictions until april 1. http://www.startribune.com/minneapolis-sets-winter-parking-restrictions-until-april-1/506388832/, 2019.

[93] Shai Ben-David et al. Measures of clustering quality: A working set of axioms for clustering. In *Adv. in neural information processing systems*, pages 121–128, 2009.

[94] PLavanya Kumari, GKrishna Reddy, and TGiridhara Krishna. Optimum Allocation of Agricultural Land to the Vegetable Crops under Uncertain Profits using Fuzzy Multiobjective Linear Programming. *IOSR Journal of Agriculture and Veterinary Science*, 7(12):19–28, 2014.

[95] Terry Van Dijk. Scenarios of central european land fragmentation. *Land Use Policy*, 20(2):149 – 158, 2003.

[96] Gajendra S. Niroula and Gopal B. Thapa. Impacts and causes of land fragmentation, and lessons learned from land consolidation in south asia. *Land Use Policy*, 22(4):358 – 372, 2005.

[97] Kai Cao, Michael Batty, Bo Huang, Yan Liu, Le Yu, and Jiongfeng Chen. Spatial multi-objective land use optimization: extensions to the non-dominated sorting genetic algorithm-II. *International Journal of Geographical Information Science iFirst*, pages 1–21, 2011.

[98] Liu Yansui, Gong Jianzhou, and Chen Wenli. Optimal land use allocation of urban fringe in Guangzhou. *China Postdoctoral Science Foundation*, 22:179–191, 2012.

[99] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[100] Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133 – 137, 1981.

[101] Thibaut Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.

[102] S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. *On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity and Applications*, pages 236–256. 1999.

[103] Piotr Berman, Bhaskar DasGupta, S. Muthukrishnan, and Suneeta Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 427–436, 2001.

[104] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *The Multiple-Choice Knapsack Problem*, pages 317–347. 2004.

[105] Using CPLEX in ampl. `http://www.ampl.com/BOOKLETS/ampl-cplex3.pdf`.

[106] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[107] 2003 northeast blackout. `https://en.wikipedia.org/wiki/Northeast_blackout_of_2003`, 2019.

[108] DTE Energy. `https://www.newlook.dteenergy.com/wps/wcm/connect/dte-web/home/service-request/common/system-improvements/tree-trimming`, 2018.

[109] Jeffrey Schweers. Hurricane Michael: Nearly 114,000 without power this morning. https://www.tallahassee.com/story/news/2018/10/10/power-outages-grow-hurricane-michael-approaches-high-winds/1591767002/, 2018.

[110] Nicole Acevedo. Puerto Rico: Single fallen tree on power line leaves 900K without power. NBC News. https://www.nbcnews.com/storyline/puerto-rico-crisis/puerto-rico-fallen-tree-power-line-leaves-900k-without-power-n865506, 2018.

[111] Emerald ash borer. `https://www.nrs.fs.fed.us/disturbance/invasive_species/eab/effects_impacts/cost_of_infestation/`, 2018.

[112] David J Nowak and Eric J Greenfield. Declining urban and community tree cover in the united states. *Urban forestry & urban greening*, 32:32–55, 2018.

[113] BBC News. Ash tree set for extinction in europe. http://www.bbc.com/news/science-environment-35876621, 2016.

[114] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

[115] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.

[116] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6517–6525. IEEE, 2017.

[117] Matti Maltamo, Jukka Peuhkurinen, Jukka Malinen, Jari Vauhkonen, Petter Packalén, and Timo Tokola. Predicting tree attributes and quality characteristics of scots pine using airborne laser scanning data. *SILVA FENNICA*, 43(3), 2009.

[118] Doo-Ahn Kwak, Woo-Kyun Lee, Jun-Hak Lee, Greg S Biging, and Peng Gong. Detection of individual trees and estimation of tree height using lidar data. *Journal of Forest Research*, 12(6):425–434, 2007.

[119] Harri Kaartinen, Juha Hyyppä, Xiaowei Yu, Mikko Vastaranta, Hannu Hyyppä, Antero Kukko, Markus Holopainen, Christian Heipke, Manuela Hirschmugl, Felix Morsdorf, et al. An international comparison of individual tree detection and extraction using airborne laser scanning. *Remote Sensing*, 4(4):950–974, 2012.

[120] Luke Wallace, Arko Lucieer, and Christopher S Watson. Evaluating tree detection and segmentation routines on very high resolution uav lidar data. *IEEE Transactions on Geoscience and Remote Sensing*, 52(12):7619–7628, 2014.

[121] Qi Chen, Dennis Baldocchi, Peng Gong, and Maggi Kelly. Isolating individual trees in a savanna woodland using small footprint lidar data. *Photogrammetric Engineering & Remote Sensing*, 72(8):923–932, 2006.

[122] Segment mean shift. http://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/segment-mean-shift-function.htm, 2018.

[123] Yiqun Xie, Guoan Tang, Shijiang Yan, and Hui Lin. Crater detection using the morphological characteristics of chang'e-1 digital elevation models. *IEEE Geoscience and Remote Sensing Letters*, 10(4):885–889, 2013.

[124] Peter Tittmann, Sohail Shafii, Bruce Hartsough, and Bernd Hamann. Tree detection and delineation from lidar point clouds using ransac. In *Proceedings of SilviLaser*, 2011.

[125] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[126] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[127] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[128] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.