UNIVERSITY OF CALIFORNIA
RIVERSIDE

Scalable and Expressive Spatial Analysis and Modeling

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Yunfan Kang

September 2023

Dissertation Committee:

Prof. Amr Magdy, Chairperson
Prof. Ahmed Eldawy
Prof. Jiasi Chen
Prof. Vagelis Hristidis

The Dissertation of Yunfan Kang is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to express my gratitude to my advisor, Professor Amr Magdy. I would not have been here without him. Professor Magdy has always been supportive and devoted. His passion twards research and education and insightful thinking guide me to broaden my eye sight towards scientific research sharpens my ability in problem solving. He spends hours and hours to revise my manuscript for each single submission and I learned a lot about story telling and academic writing through in this rewarding process. It is still a myth for me about when Professor Magdy would go to bed or take a break becasue he seems always available for me.

I sincerely thank my parents Prof. Qinma Kang and Mrs. Xiangling Kang. They show me how knowledge can change one's life and have been my role model since I was a child. My father is my first teacher in computer science and I still feel proud that I still draw inspiration from his reacher in my most recent work. My mother is the person who cares about me the most and I greatly appreciate her effort and life wisdom.

I would also thank my dissertation committee members, Prof. Ahmed Eldawy, Prof. Jiasi Chen, Prof. Vagelis Hristidis as well as Prof. Ran Wei who served as the committee member for my qualifying exam. Your insightful and encouraging advices have made my works solid in every espect.

I want to thank Professor Sergio Rey for the support and the insightful discussions in spatial analysis and regionalization. I really admire his profound knowledge in spatial data science and well as his passion towards tech and life. I hope I could be like him when I am old.

In addition, I would like to thank my labmates Yongyi Liu, Abdulaziz Almaslukh, Hussah Alrashid, and Laila Abdelhafeez. Thank you all for your help and indepensible contribution to our works. I would also thank the aluminis of UCR Geospatial Science Center for the great kindness and research insights. Let's go to the bar when we meet again.

Last but not least, I would express my great gratude to all the coauthrs for my paper. Chapter 3 of this dissertation is published in ICDE 2022. Chapter 5 is published in SIGSPATIAL 2019. Chapter 6 is published in ICDEW 2020.

I could not have completed this dissertation without the great help with all these great people aforementioned above and not mentioned.

To my parents for all the support.

ABSTRACT OF THE DISSERTATION

Scalable and Expressive Spatial Analysis and Modeling

by

Yunfan Kang

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2023
Prof. Amr Magdy, Chairperson

Spatial data analysis plays a crucial role in understanding the complex dynamics of our world across various fields including geography, urban planning, environmental science, and transportation. As spatial data becomes more intricate due to the advances in geospatial technologies, there is an urgent need for more efficient and expressive tools and techniques for its analysis and modeling. This dissertation addresses this demand, presenting a series of interconnected research contributions that have enhanced the landscape of spatial data analysis.

Firstly, the dissertation introduces the open-source Python Library *Pyneapple*. *Pyneapple* encapsulates 3 modules of algorithms for regionalization problems, machine learning techniques, and group-by queries, providing a platform for scalable and expressive spatial data analysis. Secondly, among the components of the *Pynapple*, we highlight the algorithms for solving the Expressive Max-p regions (EMP) problem and the Spatial Network Hotspot Detection (SNHD). EMP is a key component of the regionalization module of *Pyneapple*. It expands the traditional max-p regions problem by supporting multiple

constraints with SQL aggregates and range operators, which allows for more precise and enriched spatial analysis. SNHD illustrates how to identify statistically robust spatial hotspots efficiently and effectively, contributing to various practical fields, including transportation analysis and crime detection. Subsequently, it introduces "CleanUpOurworld", a comprehensive and interactive database for visualizing global litter data. This tool illuminates the global litter problem by making spatial data accessible and easy to understand at various spatial scales. Last but not least, the dissertation presents "HiDAM", a novel, research-friendly data model for high-definition maps that offers detailed and nuanced analysis of both on-road and off-road spatial data.

Collectively, these works embody a significant stride in spatial data analysis, offering a more nuanced understanding of complex spatial data and its practical applications. By developing more advanced tools and extending the existing techniques, this dissertation presents meaningful and impactful contributions to the field of spatial data analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The power of understanding and interpreting the world around us through spatial data has been a transformative force across a variety of disciplines. As we increasingly transite into a data-centric era, the significance of spatial data analysis, a multidimensional perspective that provides valuable insights into the geographic context of data, is indisputable [1–3]. With applications spanning fields as diverse as pathology [4–7], urban planning [8–10], transportation [11,12], and many others, spatial data analysis serves as the bedrock upon which critical decisions are made and policies are formulated [13–15].

The world we inhabit is not merely a flat plane but a complex matrix of interconnected physical and social systems. Our cities, environments, and transportation networks all function within the paradigm of space. The characteristics of these systems and their interrelationships are intrinsically geographical, underpinning the importance of spatial data. Therefore, understanding these entities necessitates the analysis of spatial data, an endeavor that is as vast as it is intricate.

Spatial analysis is not only becoming more prevalent, but it is also increasing in complexity, caused by the increasing data volume and the diversity of analysis tasks. On one hand, the increasing availability of digitalized spatial data, such as the data collected from geo-enabled personal devices and social networks, opens new opportunities for applications. On the other hand, emerging new technologies have also spawned new concepts and brought imagination into reality. The location of the criminals can be determined through big spatial data analysis and hotspot detection. Smart cities and autonomous vehicles can potentially bring convenience to everyone. However, the tools and techniques available for spatial data analysis often lag behind the scale and complexity of the data. Existing methods can fall short when applied to the rich, diverse, and large-scale spatial datasets now available. This highlights an urgent need for the development and enhancement of more sophisticated tools to handle, analyze, and visualize spatial data.

This dissertation delves into the above-mentioned challenge, outlining my research contributions in the realm of spatial data analysis. In the journey of research that unfolds in the pages that follow, we present an array of pivotal contributions: from an open-source library optimized for scalable and expressive spatial data analysis to innovative extensions of analytic techniques that enrich regionalization constraints. We also introduce a substantial performance improvement in hotspot detection on spatial networks, delve into the intricate task of integrating and visualizing global litter data, and lay the groundwork for high-definition data models tailored for advanced applications. Cumulatively, these elements form a cohesive narrative, illustrating how state-of-the-art advancements in spatial data analysis can not only yield meaningful insights into our complex world but also drive

informed decision-making in addressing today's pressing challenges. This introductory chapter serves to briefly underline the critical importance of spatial data and the opportunities we observe in spatial analysis, setting the stage and providing the overarching motivation for the entire dissertation.

1.1 Spatial Data and Spatial Analysis

As we enter the era of information, data has become an indefensible component of our personal life and society. Spatial data, one of the most accessible yet diverse data types, describes the distribution and attributes of phenomena that are related to a location on the surface of the earth or in a certain type of coordinate [16,17]. It provides us the basis for understanding the world around us in a geographic context and empowers applications that we could not even imagine before [18, 19]. The usage of spatial data ranges from the notification about a local weather altert that we receive on our phones to the data-driven decision-making of the countries. Spatial data is so important that the White House defines the National Spatial Data Infrastructure, which is described as "the technology, policies, standards, human resources, and related activities necessary to acquire, process, distribute, use, maintain, and preserve spatial data" [20] to improve the coordination and use of spatial data.

Spatial data are formal representations of geographic features. The geographic features are usually abstracted as points, lines, and polygons. Points represent certain events or points of interest, such as the location of crime cases and the location of restaurants. Lines connect the points in a specific order and can carry spatial properties, such as road

segments and power lines. Polygons are composed of enclosed lines and usually depict the boundary of a region, such as the boundary of a state. As a result of the complexity of the spatial data, specialized techniques are designed to store and handle these data efficiently, such as geographical information systems (GIS) [21], spatial databases [22], and spatial indices [23].

Spatial analysis is one of the key components of GIS systems and lays the foundation for utilizing spatial data [24]. It has attracted the attention of both computer science [25, 26] and social science researchers [2, 27, 28]. Originating from the map that is used to analyze the 1854 Broad Street cholera outbreak by Dr. John Snow [4], spatial analysis has gone far beyond mapping and epidemiology. Nowadays, spatial analysis allows for a deeper understanding of spatial phenomena by examining the distribution, patterns, and relationships between spatial data. As summarized by Dr. Luc Anselin [29], the spatial analysis aims to answer four fundamental questions:

1. Where do things happen
2. Why do they happen there
3. How do they affect other things and how does the context affect them
4. Where should things be located

To address these questions, spatial analysis has gone a long way to develop numerous ways to visualize, discover, and explain the patterns behind the phenomena and make use of the patterns for further optimization, regression, and prediction.

1.2 Scalability and Expressiveness: Limitation and Opportunities for Spatial Analysis

Spatial analysis is still an exciting area that is under active development [30]. Not only are we facing more difficult analysis problems as the available data volume grows exponentially, but new questions that come with new phenomena and applications. In this dissertation, we discuss our efforts in addressing the two important aspects of spatial analysis techniques: scalability and expressiveness.

Scalability, in the context of spatial data analysis, refers to the capability to efficiently handle an ever-increasing volume of spatial data. Over recent decades, the amount of spatial data has grown exponentially, putting considerable strain on existing algorithms and tools by pushing them to their operational limits. This growth is further exacerbated by the widespread use of smartphones, which not only generate a wealth of real-time geographic data but also encourage greater public participation in the creation and application of such data. Daily activities such as commuting, traveling, and social media interaction generate immense volumes of geotagged data, challenging the scalability of current data processing and storage techniques. Moreover, the real-time analysis of large geospatial datasets can be a matter of life or death during disasters, underscoring the urgent need for scalable solutions. Further complexity arises with location-based services like Uber and Google Maps, which demand real-time processing of massive spatial datasets to maintain responsiveness. Social networks also stand to benefit from scalable spatial analysis techniques that can handle real-time recommendations or anomaly detection, given the constant stream of geospatial data on these platforms. Therefore, enhancing the scalability of spatial analysis tools is not

merely a technical concern; it carries significant societal implications. Improved scalability can elevate the quality of life for individuals and communities by enabling more effective emergency response, enriching user experience, and facilitating the smooth operation of essential daily services.

Expressiveness, on the other hand, refers to the ability of a model or algorithm to accommodate various constraints, spatial relationships, and attributes. As society evolves and new challenges emerge, the expressiveness of existing spatial analysis techniques is put to the test. Take, for example, the advent of autonomous driving vehicles, a cornerstone of smart city initiatives. These vehicles require exceptionally expressive high-definition map models to facilitate complex functions like precise lane-level navigation and self-localization. Yet, most current map data, designed primarily for human interpretation rather than machine-to-machine interactions, lacks the granularity and detail to support these advanced functions. Similarly, emerging problems, such as the COVID-19 pandemic, underscore the need for algorithms with greater expressiveness. The spread of the virus is affected by a myriad of factors, ranging from population density and transportation networks to social practices. Algorithms with higher levels of expressiveness can help policymakers better understand the complexities of such problems, enabling more effective and targeted interventions. Traditional algorithms, like the max-p regions used for regional aggregation, often fall short in this context. They typically allow for just a single constraint with a sum aggregation and a lower-bound threshold, limiting their utility for complex, multi-faceted issues like pandemic management. In light of these challenges, the importance of designing spatial analysis algorithms with expressiveness in mind cannot be overstated. As the realm

of spatial problems continues to expand and evolve, so too must the capabilities of the models and algorithms we use to solve them. By doing so, we can adapt to new trends and more effectively tackle the complex spatial problems that society faces today.

1.3 Thesis Outline

In this dissertation, we present our effort to address the scalability and expressive issues in multiple aspects of spatial analysis. We first present *Pyneapple* in Chapter 2, the library for scalable and expressive spatial analysis algorithms that aims to build the bridge between social scientists and big spatial data analysis. Chapter 3 presents the expressive max-p regions problem and the heuristic algorithm under the Regionalization module of *Pyneapple* that solves this problem efficiently and effectively. Chapter 4 illustrates the scalable method for detecting statistically robust hotspots on spatial networks, which is a core component of the machine learning algorithms in *Pyneapple*. We then present CleanUpOurWorld in Chapter 5, which is a database for improving the scalability of aggregating and visualization of large-scale litter datasets. In Chapter 6, HiDaM is proposed to give an expressive and research-friendly data model for high-definition map data. Finally, we conclude the dissertation and discuss the future directions of the proposed works.

Chapter 2

Scalable and Expressive Spatial Analysis Library

2.1 Introduction

Spatial analysis has become an indispensable tool in various social science studies [1–3]. It enables social scientists to conduct complex analyses on diverse socioeconomic phenomena [31] and plays a vital role in everyday applications and policy making. For example, when the White House Council on Environmental Quality launched version 1.0 of the Climate and Economic Justice Screening Tool, the National Academies of Sciences, Engineering, and Medicine established a committee to analyze how environmental health and geospatial data can further improve the screening tool [32]. Spatial analysis is also one of the core functionalities of geographic information systems (GIS), with the ArcGIS Spatial Analyst extension being used by thousands of organizations every day.

Despite the benefits social scientists gain from spatial analysis techniques, particularly with the increased availability of big geospatial data, several emerging problems remain unsolved due to the limited scalability and expressiveness of existing tools and techniques. For example, state-of-the-art regionalization algorithm max-p-regions (MP-regions), provided in the PySAL library and ArcGIS, can take 3-10s of minutes to process a dataset with approximately 3K areas, depending on the threshold value. Consequently, real-world applications are limited to relatively small datasets, ranging from tens of areas [33–35] to only a few hundred [36–40]. Such severe scalability limitations prevent the existing MP-regions formulation from supporting a wide variety of user-defined constraints that enable users to express flexible regionalization queries for diverse applications. Additionally, existing MP-regions formulation allows a *single* user-defined *threshold lower bounded on one attribute*, e.g., total region population $\geq$ 100K. However, many analysis tasks cannot be addressed with a single constraint involving a simple threshold. For instance, studies indicate that the COVID-19 virus transmission patterns are closely related to prosperity and labor mobility [6]. Therefore, queries incorporating multiple constraints cannot be expressed using the traditional MP-regions formulation due to its limited support for user-defined constraints.

Besides regionalization, similar scalability and expressiveness challenges are faced by social scientists in other families of queries, such as learning-based spatial analysis and counting-based statistics that group spatial objects by certain spatial constraints. Therefore, improving the scalability and expressiveness of the spatial analysis techniques would benefit the existing applications and unlocks more potential for social scientists.

This chapter presents *Pyneapple*, an open-source library for scalable and expressive spatial analysis, currently under active development with more features being added. *Pyneapple* summarizes our research works, such as the expressive max-p regionalization problem in Chapter 3 and the spatial network hotspot detection problem in Chapter 4. To demonstrate the capability of *Pyneapple*, we design front-end applications so that the users can interact with the library by specifying a potential use case through interactively selecting the constraints and intuitively obtaining the corresponding visualized results. With the assistance of the GPT-3.5 model, our frontend application even allows the users to talk directly with the system and the query interpretation and processing can be handled by the backend automatically. In addition, we also provide interactive Jupyter Notebooks for both Python and Java users, which interactively demonstrate how the APIs can be invoked and the superiority of our algorithms when compared with the existing methods. A brief yet comprehensive overlook of the *Pyneapple* library is given in Section 2.2 and we introduce the potential use cases of *Pyneapple* in Section 2.3.

2.2 *Pyneapple* Overview

This section introduces *Pyneapple* library and its components. An overview of *Pyneapple* is given in Figure 2.1. *Pyneapple* comprises three main modules: regionalization queries, machine learning (ML) assisted analysis, and group-by aggregation queries. Each module offers well-documented Python and Java APIs to be seamlessly integrated with the data science ecosystem. The remaining of this section introduces the modules and the algorithms they contain.

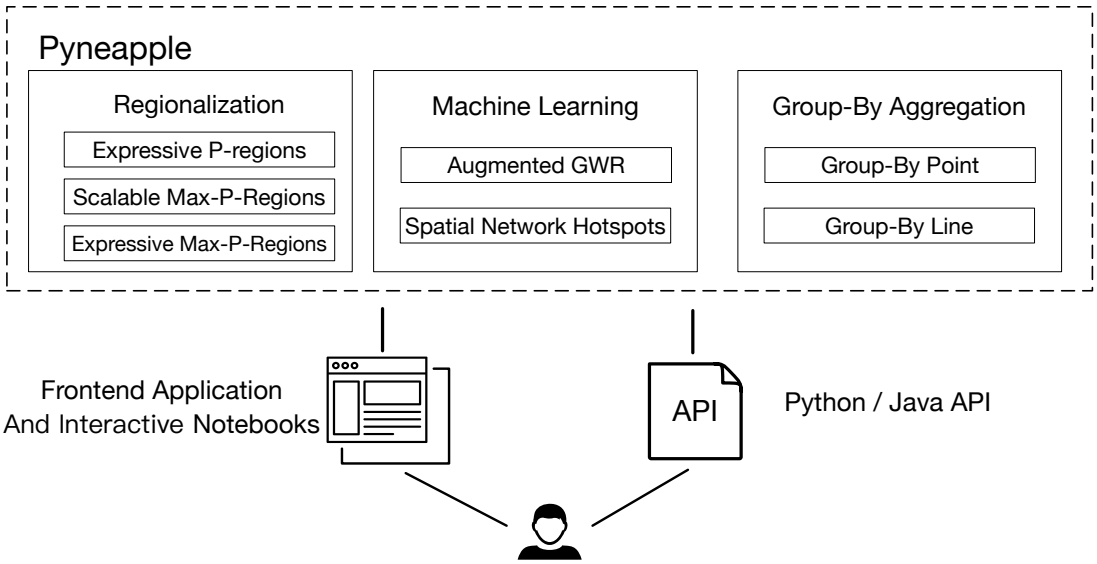


Figure 2.1: *Pyneapple* Overview

2.2.1 Spatial Regionalization

Spatial regionalization is the process of grouping a set of spatial areas into spatially contiguous and homogeneous regions. This module contains three algorithms: the scalable p-regions with user-defined constraints (PRUC) [41], the scalable MP-regions algorithm (SMP) [42], and the expressive MP-regions algorithm (EMP) [43].

The p-regions problem is similar to the clustering problems, such as k-means, where the users need to specify the number of regions and an optimization objective. The **PRUC** problem [41] generalizes the p-regions problem to support an additional user-defined constraint, such as the minimum population of each region. The Global Search with Local Optimization algorithm is designed to solve PRUC, boasting a 100× speed increase, and supports 4× larger datasets than the state-of-the-art algorithms.

Different from the p-regions problem, the MP-regions problem asks the users to give a constraint on an attribute with a lower bound threshold, such as the minimum total

population, to meaningfully define the spatial scale of the regions instead of specifying the number of regions. This is widely used to automatically discover the spatial scale when the social scientist cannot manually specify the number of regions p, e.g., p=50 that is used for state-level phenomena is very different from p=3000 that is appropriate for county-level phenomena in USA. However, scalability issues hinder its use with a large number of geographical units [39, 44, 45]. To address this, **SMP** [42] optimizes the region-building strategy by tuning the randomized area selection, achieving a 97% reduction in query time and supporting datasets that are an order of magnitude larger than the state-of-the-art approaches. In addition to the scalability issue, the MP-regions formulation is also lacking in expressiveness because it only supports one constraint with a lower-bound threshold. In order to support a broader range of complex analysis tasks, **EMP** [43] is proposed and adopts the five SQL aggregation operators and supports multiple constraints in a single query. Each constraint can be specified with a range operator, which can also represent lower bounds and upper bounds. A two-phase greedy algorithm is implemented to solve the novel EMP formulation and demonstrates great scalability when processing datasets that are larger than the ones used in the existing literature. Detailed discussion about the EMP problem and the corresponding algorithm is presented in Chapter 3.

2.2.2 ML-Assisted Spatial Analysis

The machine learning (ML) assisted spatial analysis consists of two algorithms: Augmented Geographically Weighted Regression (A-GWR) [46] and ML-assisted local hotspot detection.

A-GWR surpasses the limitations of Geographically Weighted Regression (GWR) in terms of scalability and expressiveness by employing Stateless Multilescale GWR (S-MGWR), which allows MGWR to efficiently train different data features in parallel to influence different spatial scales with independent spheres of influence. A-GWR enables variants of GWR to utilize parallelized black-box optimizations to determine optimal parameters. In addition, A-GWR boots expressiveness through incorporating versatile general-purpose ML models to achieve better flexibility and a huge performance boost. Compared to GWR and MGWR, A-GWR achieves higher accuracy, expressiveness, and is up to 14.4 times faster.

The second module aims to improve the scalability of the existing **local hotspot detection on spatial networks** [47]. Existing clustering methods are criticized for lacking statistical significance, while statistical methods suffer from serious scalability issues. To avoid false positives and produce statistically robust results, we adopt the local network K-function (LocalNK), a variant of the well-known Ripley's K-function [48] in the network space. The state-of-the-art LocalNK methods [49, 50] struggle with scalability and cannot determine the scale of the hotspots without brute force enumeration. To efficiently enumerate all potential hotspot subnetworks, we propose a prune-and-refine with proof of correctness and completeness. To determine whether the enumerated subnetworks are hotspots or not, we design novel network local exploration strategies and graph embedding techniques that exploit the potential of machine learning classifiers and deep graph neural networks. In addition, the optimized modules for preprocessing are also included in *Pyneapple* for effective map matching and dataset compression. With multiple modules that cover the whole procedure, the hotspot detection model efficiently processes raw data from the

network and events, returning all statistically robust hotspots. Further details about local hotspot detection on spatial networks are available in Chapter 4.

2.2.3 Group-By Queries

Grouping spatial units, such as points and line segments, are widely used in different spatial analysis operations, e.g., polygonization or counting-based spatial statistics. The point group-by query is a composition of the fundamental spatial range query using sets of polygons. Answering such queries with real datasets that contain hundreds of millions of data points and real polygons with millions of points on the perimeters is highly inefficient. The query latency can easily escalate to hours of processing on a dozen CPUs in those cases. Our **Spatial GroupBy Polygon Aggregate Counting (SGPAC)** module [51,52] is designed to integrate seamlessly with mainstream spatial data management systems, providing highly-parallelized processing capabilities for spatial point group-by queries, reducing query latency by up to an order of magnitude compared to existing competitors. The line group-by query discovers planner spatial polygons out of large datasets of line segments, such as large-scale road networks. Our **Distributed Doubly-Connected Edge List (DDCEL)** module [53] provides the first parallelized algorithm to build DCEL data structure efficiently at a large scale to discover planner polygons from scattered line segments. DCEL is widely used to support fundamental spatial operations, such as spatial overlay for multi-layer datasets.

2.3 *Pyneapple* Use Cases

Pyneapple provide various user interfaces for different groups of target users. Users will interact with the *Pyneapple* library through user-friendly web applications and Jupyter Notebooks, illustrating the ease of use and interactive nature of the toolkit. Through the interactive web UI, users are able to visualize the results, explore different parameter settings, and gain a deep understanding of the potential applications of *Pyneapple* in various domains. After gaining experience with the vivid and interactive use cases, users will be confident in the capabilities of *Pyneapple* to address complex spatial analysis tasks and appreciate the potential impact it can have on real-world decision-making processes. Following this, *Pyneapple* encourages the users to make use of the APIs in Python and Java by providing example notebooks to prove that algorithms included in *Pyneapple* can be seamlessly integrated into their existing workflow.

2.3.1 Use Case 1: Interactive Querying and Visualization

To demonstrate its capabilities and present the core functionality of *Pyneapple* to users with varying technical backgrounds in a straightforward way, we have developed an intuitive and user-friendly frontend that offers a straightforward web UI. Users can easily issue queries by selecting datasets and composing constraints by interacting with dropdown menus and sliders. For example, Figure 2.2 showcases the UI for the regionalization algorithms. Let's assume a user is interested in exploring the employment situation in Los Angeles. By clicking the *Browse* button, the shapefile of the census tracts of Los Angeles is loaded and visualized in the left panel. Simultaneously, the contiguity matrices are

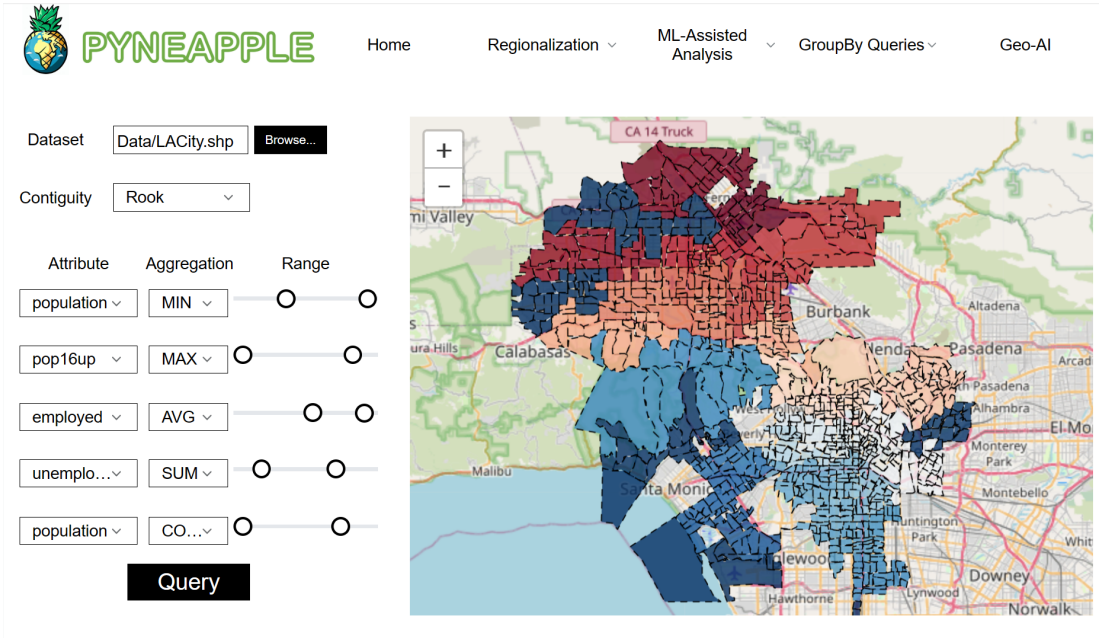


Figure 2.2: Frontend UI for Regionalization Queries

computed at the back end so that the user can select how he wants to define the connectivity of the regions. After specifying the attribute names and the aggregation methods, the slide bars are initialized, allowing users to set the range for each constraint intuitively. Upon pressing the *Query* button, the query is sent to the backend, and the corresponding regionalization algorithm (in this case, the EMP module) performs the regionalization and returns the region labels. The regions are then automatically colored, and the users can interact with the visualization to view the detailed attribute values by clicking on the regions. For use cases whose scalability is greatly improved by *Pynepple*, such as the SMP, hotspot detection, and the group-by queries, we also include the APIs of the state-of-the-art implementations. The frontend application also provides an option to turn on the side-by-side comparison with statistics to illustrate the superiority and the correctness of *Pyneapple* algorithms. This example use case highlights the potential of *Pyneapple* in making spatial

analysis more accessible and efficient for various users, emphasizing its ability to handle large-scale problems and produce accurate results.

2.3.2 Use Case 2: Building Applications

Pyneapple provides Application Programming Interfaces (APIs) with two levels of detail to cater to different user needs. For social scientists working primarily in the Python environment, *Pyneapple* encapsulates all low-level implementations as black boxes and exposes just the essential functions as plug-and-use modules. Although several modules are originally implemented in Java or rely on Java libraries, *Pyneapple* encapsulates all Java APIs and provides Python APIs that comply with the specifications of the existing libraries and systems.

To demonstrate the simplicity of *Pyneapple* Python APIs, we carefully design Jupyter Notebooks for each of the modules [54]. Each notebook contains detailed guidance and examples to walk the user through the life cycles of algorithms in *Pyneapple*. To get started, we show that social scientists can simply install the *Pyneapple* Python package through pip. Detailed All dependencies, including the JDK and the configuration of the Java environment, can be configured by the script and the users will not be aware of the Java-Python bridge. For demonstrating the usage of *Pyneapple* Python APIs, we adopt familiar, classic, or popular use cases that social scientists can relate to in order to demonstrate how the APIs of the existing techniques can be replaced with *Pyneapple* APIs that accept the same input and produce compatible output within the original computation workflow. Comparisons between *Pyneapple* APIs and original APIs, along with corresponding visualizations, are shown to highlight *Pyneapple*'s superiority in terms of efficiency and

result quality. Through this example, we show users that their work is painlessly improved with *Pyneapple* APIs while gaining huge benefits in scalability and expressiveness.

Java-based APIs with modularity design and detailed documentation are also provided to exploit better the potential of the *Pyneapple* algorithms. To demonstrate this, we design another set of notebooks with the help of *IJava* kernel. These Java notebooks demonstrate the low-level APIs in great detail so that the developers may customize individual steps to adapt the algorithms to new problems or perform fine-tuning as needed. For instance, users can modify the original contiguity functions in the regionalization module to define adjacency differently, considering two remote areas as neighbors to produce non-contiguous regions. Once this step is modified, the audiences can track the impact on the remaining module, such as the spatial partition behavior of the SMP algorithm or the multi-constraint construction phase of the EMP algorithm. Moreover, users will also have full access to the logs of the parallel computations, such as the usage of *Spark* in the group-by queries. With access to context and logs, they can fine-tune configurations and receive feedback on performance changes after reaching the breakpoint. This level of customization and control ensures that developers can fully harness the power of the *Pyneapple* library to tackle a wide range of spatial analysis challenges.

2.3.3 Use Case 3: Natural Language Querying

To further expand the use cases of *Pyneapple* and extend its interactivity, we introduce a middle layer powered by GPT-3.5 APIs to enable the audiences to interact with the system using natural language. This innovative approach allows users to either

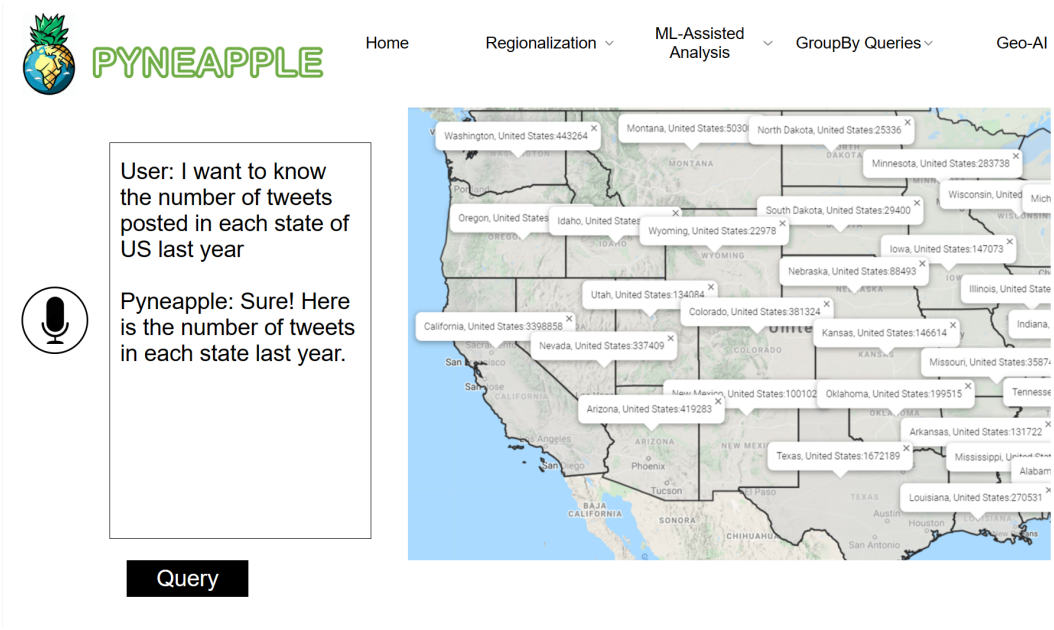


Figure 2.3: Query by Talking to the System

speak their query or type it in as text, making the process more user-friendly and intuitive. For example, a user may say, "I want to know the number of tweets posted in California last year", as shown in Figure 2.3 The recorded audio is then processed by the large-v2-whisper model to generate text. With the help of the text-davinci-003 model, the AI-driven system would recognize the query, identify the required dataset and parameters to invoke the corresponding function, and execute the necessary steps to visualize the results on the frontend. The user can modify the query to get a more desirable result or issue another query by simply continuing the discussion.

Chapter 3

Max-P Regionalization with Enriched Constraints

3.1 Introduction

Regionalization is the process of clustering a set of areas into spatially contiguous and homogeneous regions. Unlike traditional clustering of multi-dimensional points, the basic units in regionalization are spatial polygons that are grouped based on both attribute values and adjacency in space. Regionalization is widely used in numerous applications such as epidemic analysis [6], service delivery systems [55], water quality assessment [56], weather temperature classification [57], rainfall erosivity estimation [58], health data analysis [59], spatial crowdsourcing [60, 61], and constituency allocation [62]. Identifying regions is a computational and conceptual challenge. One of the fundamental challenges is determining the spatial scale of the studied phenomena. For example, *snowing* is spatially correlated

at the level of multiple states, e.g., the USA midwest states, while *temperature* is spatially correlated at a county level. Most regionalization routines require the number of regions, p, as an input parameter, e.g., in the USA, $p = 50$ is a state level and $p = 3000$ is a county level. This puts the burden of determining the spatial scale on the data analysts, forcing them to analyze multiple p values, which increases the computational cost and limits the query expressiveness.

The latest regionalization formulation, called the *max-p-regions* problem (or MP-regions), has addressed the scale problem [63] and become popular in different applications, including urban spatial structures [36], measuring intra-urban poverty [37], population growth analysis [33], crime analysis [38], regional inequality analysis [34], spatial uncertainty [64], and neighborhood delineation [40]. With the observation that researchers usually know a condition that a region must satisfy to be suitable for the analysis, the MP-regions automatically discovers the maximum number of regions that satisfy a given user-defined constraint instead of forcing users to input p. However, the MP-regions problem has several limitations in both scalability and expressiveness. Currently, the state-of-the-art algorithms take 3-60 mins to process a dataset with ∼3k areas, depending on the threshold value. As a result, existing real applications, listed above, use relatively small data, ranging from tens of areas [33–35] to only a few hundred [36–38, 40, 64]. It is explicitly stated in [64] that to use the MP-regions algorithms "*one must be willing to reduce the number of geographic units of analysis.*" Such severe scalability limitations prevent the existing MP-regions formulation from supporting a wide variety of user-defined constraints that enable users to express flexible regionalization queries for various applications.

Existing MP-regions formulation allows a *single* user-defined *threshold lower bound on one attribute*, e.g., total region population $\geq$ 100K. However, many analysis tasks cannot be catered with a single constraint that involves a simple threshold. In many real applications, it is common to have multiple constraints with different aggregate functions. For example, studies show that the COVID-19 virus transmission pattern is closely related to prosperity and labor mobility [6]. In order to make region-specific recommendations for preventing the virus spread, policymakers can issue a query to identify reasonably populated regions with a total population $\geq$ 200000, an average monthly income between \$3000 to \$5000, and public transportation passengers $\geq$ 10000. Another example is studying the changes in population requires considering multiple factors that influence the phenomenon [33] with different aggregation functions such as the *minimum* population of each area, the *maximum* school drop-out rate, the *average* age of the population, and *total* unemployment. A third example is the patrol sector partition problem [35] that aims to consider the number of calls and the workload balance. These example applications significantly benefit from enriching the expressiveness of MP-regions queries that will enable exploring different scenarios and expanding analysis capabilities. Currently, such advanced regionalization queries cannot be expressed using the existing MP-regions formulation due to its limited support for user-defined constraints, limiting its full potential in many real applications.

This chapter introduces the enriched MP-regions problem (EMP) that addresses the existing MP-regions limitations, the third algorithm included in the Regionalization module of the *Pyneapple* library. EMP enables users to analyze one to two orders of magnitude larger datasets than the currently supported ones. Furthermore, EMP enriches

user-defined constraints with three novel features: (1) EMP provides an enriched set of aggregates that contains three families of aggregate functions, namely, *extrema aggregates*, *centrality aggregates*, and *counting aggregates*. The three families of aggregates are inspired by the standard SQL aggregate functions from which we adopt minimum and maximum as extrema aggregates, average as a centrality aggregate, and summation and count as counting aggregates. (2) EMP allows using a range comparison operator that expresses all possible comparisons, less-than-or-equal ($\leq$), larger-than-or-equal ($\geq$), and range comparison. This enables users to set both lower and upper bounds for their constraints instead of only lower bounds in existing formulations. (3) EMP supports multiple constraints instead of a single constraint. The EMP formulation addresses the limitation of the existing MP-regions formulation and reveals the full potential for regionalization queries in different applications.

The EMP problem is an NP-hard problem (see Section 3.4), so finding an optimal solution is intractable. We formulated the EMP as a mathematical optimization problem using mixed-integer programming (MIP) problem [65] and solved it with the world's fastest MIP solver Gurobi [66]. The optimizer takes 33.86s to find an optimal solution for 9 areas, 10 hours for 16 areas, and is not able to return a feasible solution for 25 areas after 110 hours of execution [67]. This clearly shows the infeasibility of finding an exact solution for hundreds or thousands of areas in a reasonable runtime. However, even finding an approximate solution faces two main challenges. First, the newly introduced constraints are non-monotonic. Therefore, adding or removing areas to and from a certain region does not provide any guarantee of satisfying or violating these constraints. This requires exhaustive

exploration for the whole search space to find the solution, which is intractable with such an NP-hard problem. Second, while satisfying multiple non-monotonic constraints, a region can easily become overlarge. Having oversized regions reduces the number of regions p, which contradicts the objective of maximizing p.

To address these challenges, we introduce $FaCT$, a three-phase algorithm to solve the EMP problem. The first phase derives theoretical bounds to verify the feasibility of finding a solution given the user-defined constraints. In addition, in the case of infeasibility, it enables the user to potentially alter input data to satisfy the constraints. This provides great flexibility to tune either data or query parameters adaptively and enables rich exploratory analysis capabilities for a wide variety of datasets. The second phase constructs a feasible solution that satisfies the input constraints while maximizing the number of regions p. The phase is divided into three independent steps with the following features: (a) Each step is carefully designed to exploit the mathematical properties of one type of constraint to maximize the p to its theoretical upper bound. (b) The input and output of each step are designed exquisitely to reduce the search space for the following steps and mitigate the conflicts introduced by the different mathematical properties of different constraints. (c) The steps are independent of each other, enabling $FaCT$ to construct feasible solutions progressively and handle any arbitrary combinations of constraint types effectively and efficiently. (d) The algorithm provides multiple parameter tuning options so that the users can choose the best aggregate strategy according to the use case and the characteristic of the dataset. (e) The algorithm supports datasets with multiple connected components, while the MP-regions formulation supports only a single connected component. The third phase

of $FaCT$ performs a local search adapting the Tabu search [68] and swaps areas between regions to improve the overall heterogeneity of the regions.

Our experimental evaluation demonstrates the effectiveness of $FaCT$ on real datasets. We evaluate the impact of different threshold ranges, different combinations of constraints, and the scalability to process datasets. Our contributions are summarized as follows:

- We define an enriched max-p regions problem (EMP) that reveals the full potential of max-p regionalization.
- We prove the NP-hardness of the EMP problem.
- We propose $FaCT$; a three-phase greedy algorithm that provides approximate solutions for the EMP problem.
- We perform an extensive experimental evaluation that shows the effectiveness of our techniques on real datasets.

The rest of this chapter is organized as follows. Section 3.2 presents the related work. Section 3.3 defines the problem and Section 3.4 proves its NP-hardness. Our proposed solution is detailed in Section 3.5, and its time and space complexity are analyzed in Section 3.6. Section 3.7 shows the experimental evaluations and Section 3.8 concludes the chapter.

3.2 Related Work

The regionalization problem originates from the demand for aggregating homogeneous regions in the analytical tasks performed by social scientists and statisticians [69]. Regionalization is referred to by different names, including region building [70], conditional clustering [71], clustering with relational constraints [69], contiguity-constrained clustering [72], constrained classification [73], maximum split clustering under connectivity constraints [74], and p-regions problem [75] that is used in a variety of applications such as detecting functional regions [76, 77], network-constrained urban management [78], and partitioning compact regions [79–81]. The max-p-regions problem (MP-regions) [63] has been introduced to eliminate the requirement for inputting the number of regions in advance. It aggregates areas into a maximum number of regions so that each region satisfies a single user-defined constraint with a summation aggregate, e.g., total population $\geq$ 20K. Existing variants [33, 82, 83] use the road network-based connectivity as an additional spatial constraint to aggregate regions or use multiple attributes to impose the constraint. However, all existing variants still support only a single constraint of one type as the MP-regions problem.

Existing regionalization techniques that perform multi-criteria districting [33, 35, 84–86] use multi-objective optimization to partition the space based on multiple attributes. These techniques still put the burden of determining the spatial scale, i.e., the number of output regions, on the user. They also allow the user to define a single constraint on one of the attributes, limiting the query expressiveness. Our EMP problem addresses these limitations. It enables users to define multiple constraints on different attributes to

provide flexible and expressive queries. In addition, it inherits the automatic discovery of the appropriate number of regions from the MP-regions and eliminates the spatial scale problem.

We classify the popular approaches to tackle the NP-hard rationalization problems into two main categories. The first category is the clustering methods [87, 88] with two phases. The first phase clusters the centroids of polygons using a traditional clustering algorithm. The second phase then imposes the spatial connectivity constraint to ensure that each region is geographically connected. The second category adopts a two-phase contiguity-constrained heuristic optimization approach [63,83,89–91]. The set of spatial areas is usually represented by a graph that encodes the spatial contiguity relationships. With the spatial contiguity constraint imposed explicitly, a feasible initial solution is constructed in the first phase and then optimized using heuristic or mixed-heuristic search methods in the second phase. The construction methods include tree partition [89,92] and greedy aggregation [63, 82,90,91]. The heuristic search phase optimizes the objective function to improve the region homogeneity [89, 90, 92] or geographical compactness [90] without violating the contiguity constraint.

Our work follows the second category of contiguity-constrained heuristic optimization search. We consider multiple constraints with different aggregate functions that do not exist in the literature, so none of the existing methods can obtain a feasible solution that satisfies our enriched constraints.

3.3 Problem Definition

This section formally defines the enriched max-p regions (EMP) problem. Let $A = \{a_1, a_2, ..., a_n\}$ denote the set of areas. Each area a_i is defined by four attributes: a_i=(i,b_i,S_i,d_i), where i is the area identifier, b_i is an arbitrary spatial polygon that defines the area's spatial boundaries, S_i is a set of spatially extensive attributes and d_i is a dissimilarity attribute. The dissimilarity attribute d_i is used in computing output regions' heterogeneity. The rest of the section presents basic definitions, then defines the EMP problem formally.

Definition 3.3.1 (User-defined Constraint)**.** A user-defined constraint c is a condition that is defined as "$l \leq f(s) \leq u$" or "$f(s) \in [l, u]$". c is identified with a 4-tuple: (f,s,l,u), where f is an aggregate function, s is a spatially extensive attribute, $l \in [-\infty, \infty)$ is a number that represents a lower bound, and $u \in (-\infty, \infty]$ is a number that represents an upper bound.

EMP allows f to be MIN, MAX, AVG, SUM, or COUNT that compute minimum, maximum, average, summation, and count aggregates, respectively, with the same semantic as the SQL standards. The attribute s is a spatially extensive attribute whose value is divided over the smaller areas when the area is fragmented. For example, the *population* value of a state is divided over its cities, so the *population* is a spatially extensive attribute. This is the opposite to spatially intensive attributes, such as *temperature*, that are not divided when the spatial area is fragmented. Examples of s include the population, labor force, and households in each area. When $l = -\infty$, the range has an open-ended lower bound and the constraint becomes $f(s) \leq u$. Similarly, when $u = \infty$, it becomes $f(s) \geq l$.

Definition 3.3.2 (Region)**.** A region R is a set of g areas: $R = \{a_1, a_2, ..., a_g\}$, such that: (1) $g \geq 1$, i.e., R contains at least one area. (2) Areas in R are spatially contiguous,

i.e., $\forall\ a_i, a_j \in R$, $\exists$ a sequence of areas (a_k,...,a_l) s.t. both (a_i, a_k) and (a_l, a_j) are spatial neighbors, and every two consecutive areas in the sequence are spatial neighbors.

Definition 3.3.3 (Heterogeneity)**.** Heterogeneity $H(P)$ of a set of regions P is defined as:

$$H(P) = \sum_{\forall R \in P} \sum_{\forall a_i, a_j \in R} |d_i - d_j| \tag{3.1}$$

This definition of H is popular in the MP-regions literature, so it is recognized by social sciences experts as a primary measure for region heterogeneity. However, our work can support alternative definitions, such as improving spatial compactness or balancing multiple criteria. The reason is that our second phase, which is based on Tabu search as discussed in Section 3.5, can deal with different optimization functions.

EMP Problem. The EMP problem is defined as follows:

Input: Given: (1) A set of n areas: $A = \{a_1, a_2, ..., a_n\}$. (2) A set of user-defined constraints $C = \{c_1, c_2, ..., c_m\}$.

Output: (1) A set of regions $P = \{R_1, R_2, ..., R_p\}$, where $1 \leq p \leq n$ and each region R_i satisfies the below EMP constraints and objectives. (2) A set $U_0 = A - \bigcup_{i=1}^{p} R_i$, i.e., U_0 contains all areas that are not assigned to any feasible region R_i, $\forall 1 \leq i \leq p$. Areas in U_0 may not be spatially contiguous.

EMP Constraints:

- $R_i \cap R_j = \emptyset, \qquad \forall R_i, R_j \in P \wedge\ i \neq j$
- R_i satisfies all constraints $c_j \in C$,

$$\forall\ 1 \leq i \leq p, 1 \leq j \leq m$$

Objectives:

- Maximizing the number of regions p.
- Minimizing regions' overall heterogeneity $H(P)$.

EMP has two objectives. In case they contradict during building the output regions, the first objective (maximizing the number of regions) is favored over the second one (minimizing regions' heterogeneity). This allows users to obtain the maximum number of desirable regions without providing the number of regions as an input, as favored by the domain experts [63], which addresses a major limitation in the previous regionalization problems. The second objective uses a dissimilarity attribute that is not necessarily a spatial attribute. For example, social scientists produce regions that are homogeneous in average income level.

Comparing the EMP problem with the original MP-regions problem [63], there are two major differences that are introduced by the enriched user-defined constraints. First, the enriched constraints are not always monotonic. Assuming that all spatially extensive attribute values are positive, adding an area to a region or removing an area from a region in MP-regions guarantees a monotone change towards satisfying the constraint threshold. This is not the case for the EMP problem due to allowing AVG, MIN, and MAX constraints. In addition, the EMP problem allows upper-bounded threshold ranges, so adding areas to a region without additional validation leads to violating the SUM and COUNT constraints' upper bounds. Second, the EMP problem allows unassigned areas, U_0, to exist in the final area partition, contrary to the MP-regions problem. This provides flexibility to satisfy multiple constraints of different types, especially MIN and MAX constraints that commonly

cause invalid areas, as discussed in the following sections. Allowing unassigned areas enables constraints such as MIN and MAX to serve as filters to filter out areas with undesired properties, which introduces more tolerance and enables richer exploration capabilities to regionalization queries.

3.4 NP-hardness of EMP

This section proves the NP-hardness of the EMP problem. The NP-hardness of the EMP problem follows the results that the MP-regions problem is NP-hard [63, 93, 94]. The MP-regions problem takes as input: (1) a set of spatially contiguous areas $A = \{a_1, a_2, ..., a_n\}$. Each area a_i is associated with an identifier i, a spatial polygon b_i, a spatially extensive attribute s_i and a dissimilarity attribute d_i, (2) a threshold value t. For the purpose of the proof, we solve both the MP-regions and the EMP problem as decision problems with a fixed $p = k$.

Theorem 1. *The EMP problem is NP-hard.*

Proof. Let X = (A, t) be an instance of the MP-regions problem. We construct an instance Y = (A', C) of the EMP problem as follows: (1) For each area $a_i \in X.A$, add to $Y.A'$ an area $a'_i = \{i, b_i, S'_i, d_i\}$, $S'_i = \{s_i\}$. (2) Let $C = \{(SUM, S'[0], t, \infty)\}$. (3) Let the number of regions p equal k. This reduction is of polynomial time of $O(n)$ where n is the number of areas in A. Therefore, an algorithm that decides on the instance Y of the EMP problem decides on the instance X of the MP-regions problem as well. As the MP-regions problem is NP-hard, then the EMP problem is NP-hard, and the proof is complete. □

3.5 Proposed Solution

This section presents *FaCT*, a three-phase algorithm that finds a feasible solution for an instance of the EMP problem with low overall heterogeneity. The first phase is a ***feasibility phase*** that verifies the existence of any feasible solution given the user-defined constraints. The second phase is a ***construction phase*** that constructs a feasible initial solution with the maximum number of regions p while satisfying all user-defined constraints. This phase represents the major contribution of this algorithm. It balances two challenging objectives: satisfying multiple non-monotonic user-defined constraints and producing a maximum number of spatially contiguous regions. The third phase is a ***local search phase*** that improves the initial solution of the *construction phase* in terms of the heterogeneity score. The following sections detail each phase.

3.5.1 Feasibility Phase

The EMP problem deals with arbitrary combinations of constraints on different attributes. The goal of the feasibility phase is to signal the user at an early stage for: (1) the infeasibility of finding a solution for the given set of constraints on the given dataset, and (2) the possibility of removing some areas to satisfy the given constraints.

The fact that our algorithm can detect and filter out invalid areas for all constraints, including non-monotonic constraints, gives it major practical advantages when employing multiple constraints. The algorithm is not only able to give users a heads-up on the infeasibility of their constraints, but also users can choose to remove input areas that

cause such infeasibility automatically. This gives data analysts great flexibility to analyze a wide variety of datasets and constraints.

Given a set of user-defined constraints $C = \{(MIN, s_{min}, l_{min}, u_{min}), (MAX, s_{max}, l_{max}, u_{max}), (AVG, s_{avg}, l_{avg}, u_{avg}), (SUM, s_{sum}, l_{sum}, u_{sum}), (COUNT, s_{count}, l_{count}, u_{count})\}$, we check the feasibility of applying each type of constraint as follows:

(1) **For AVG constraints**, we compute the average value AVG(s_{avg}) of attribute s_{avg} over all areas. If $AVG(s_{avg}) < l_{avg}$ or $AVG(s_{avg}) > u_{avg}$, we infer that there exists no partition where all regions satisfy the AVG constraint without removing any area. This is justified by Theorem 3 below.

Theorem 2. *If $\exists P = \{R_1, R_2, ..., R_p\}$ so that every region $R_i \in P$ satisfies $c = (AVG, s_{avg}, l_{avg}, u_{avg})$ and P contains all areas in a set A, then the average value $AVG(s_{avg})$ of s_{avg} over all areas of A satisfies c, i.e., $l_{avg} \leq AVG(s_{avg}) \leq u_{avg}$.*

Proof. If $p = 1, i.e., P = \{A\}$, the proposition is trivially true. If $p > 1$, as all regions in P satisfy c, we have:

$$l_{avg} \leq R_i.AVG(s_{avg}) \leq u_{avg}, \forall i, 1 \leq i \leq p \tag{3.2}$$

where $R_i.AVG(s_{avg})$ represents the average value of attribute s_{avg} over areas in region R_i. Let $|R_i|$ denote the number of areas in region R_i. As P contains all areas, we have:

$$\sum_{i=1}^{p} |R_i| = n \tag{3.3}$$

Also, by definition:

$$AVG(s_{avg}) \times n = \sum_{i=1}^{p}(|R_i| \times R_i.AVG) \tag{3.4}$$

$$\begin{aligned} AVG(s_{avg}) &= \frac{\sum_{i=1}^{p}(|R_i| \times R_i.AVG)}{n} \\ &\leq \frac{\sum_{i=1}^{p}|R_i|u_{avg}}{n} \;=\; u_{avg} \end{aligned} \tag{3.5}$$

$$\begin{aligned} AVG(s_{avg}) &= \frac{\sum_{i=1}^{p}(|R_i| \times R_i.AVG)}{n} \\ &\geq \frac{\sum_{i=1}^{p}|R_i|l_{avg}}{n} \;=\; l_{avg} \end{aligned} \tag{3.6}$$

Equations 3.5 and 3.6 $\implies l_{avg} \leq AVG(s_{avg}) \leq u_{avg}$.

Hence, the Theorem 2 is true, and the proof is complete. □

Theorem 3. *For an area set A and an AVG constraint $c = (AVG, s_{avg}, l_{avg}, u_{avg})$, if the average value of s_{avg} over A, $AVG(s_{avg})$, does not satisfy c, i.e., $AVG(s_{avg}) < l_{avg}$ or $AVG(s_{avg}) > u_{avg}$, then $\nexists P = \{R_1, R_2, ..., R_p\}$ such that P partitions all areas in A and $R_i \in P$ satisfies c, $\forall 1 \leq i \leq p$.*

Theorem 3 is proved by proving its contrapositive Theorem 2.

(2) **For MIN constraints**, we compute the minimum and maximum values over all areas for attribute s_{min}, $MIN(s_{min})$ and $MAX(s_{min})$, respectively. Two different cases could cause infeasibility: (a) If $MAX(s_{min}) < l_{min}$ or $MIN(s_{min}) > u_{min}$, then there is no area that satisfies the MIN constraint and no valid regions can be constructed, so there is no feasible solution. (b) If $MIN(s_{min}) < l_{min} < MAX(s_{min})$, all areas with $s_{min} < l_{min}$ are invalid areas that cannot be part of any valid region, so they must be filtered out to

build a feasible solution. Therefore, there is room to find a feasible solution after removing invalid areas.

(3) **Similarly for MAX constraints**, there are two cases of infeasibility: (a) If $MIN(s_{max}) > u_{max}$ or $MAX(s_{max}) < l_{max}$, there is no area that satisfies the MAX constraint, and there is no feasible solution. (b) Areas with $s_{max} > u_{max}$ are invalid areas that must be filtered out to find a solution.

(4) **For SUM constraints**, we compute the minimum and summation of all areas for s_{sum}, $MIN(s_{sum})$ and $SUM(s_{sum})$, respectively. There are three cases of infeasibility: (a) If $MIN(s_{sum}) > u_{sum}$, no region can have s_{sum} within the range. (b) If $SUM(s_{sum}) < l_{sum}$, even the trivial region containing all areas cannot have s_{sum} within the range. (c) Areas with $s_{sum} > u_{sum}$ are invalid and must be removed.

(5) **For COUNT constraints**, if the number of areas $n < l_{count}$, there is no feasible solution as a region containing all areas cannot meet the lower bound l_{count}.

The feasibility phase iterates through the area set and computes the needed attribute aggregates for all constraints in a single pass. This pass is enough to filter out invalid areas, that have $s_{min} < l_{min}$, $s_{max} > u_{max}$, or $s_{sum} > u_{sum}$, to eliminate potential invalid regions. However, the feasibility pass is not enough to filter out invalid areas for the AVG constraint. This is performed during the following construction phase while imposing AVG constraints. All invalid areas are filtered out from A, added to the set U_0, and they are not considered in any further processing. The remaining areas in set A are passed to the following phases.

3.5.2 Construction Phase

The construction phase greedily constructs a feasible solution that: (a) satisfies all user-defined constraints, (b) maximizes the number of regions p, and (c) minimizes the number of unassigned areas. This phase faces the challenges of simultaneously satisfying multiple non-monotonic constraints while preventing building oversized regions that contradict maximizing p. To address these challenges, the construction phase satisfies each family of user-defined constraints in a separate step, and each step handles different constraints in isolation. By this, we gain several benefits towards addressing the challenges. First, each step focuses on one type of optimization, making it an easier problem to solve. Second, dividing the logic into consecutive steps provides flexibility to handle queries with any arbitrary subset of different constraints (as discussed in Section 3.5.4). Third, the order of steps makes use of the mathematical properties of different aggregate functions to increase the probability of producing a feasible solution. MIN, MAX, and SUM constraints are used as filters to oust invalid areas and provide seed areas for the following steps. Then, AVG constraint, which is the most challenging to satisfy, takes the seed areas and grows as many regions as possible. The last step performs only necessary modifications to satisfy any violated SUM and COUNT constraints, which are easier to satisfy due to the monotonicity of SUM and COUNT aggregate functions. Fourth, separated steps facilitate maximizing the value of p, as detailed later, without violating the constraints that have been satisfied previously. Consequently, our construction phase produces an initial partitioning for the space with a near-optimal p value that is fed to the following phase to minimize the regions' heterogeneity. Although reducing the overall heterogeneity score is an objective for the final

solution, this phase is not primarily designed to minimize heterogeneity, which is optimized in the following phase.

The construction phase executes multiple iterations. Each iteration produces a feasible partition, and we maintain the partition with the highest p value. Each iteration is divided into three steps. The first step satisfies extrema constraints (MIN and MAX constraints), the second step satisfies centrality constraints (AVG constraints), and the third step satisfies counting constraints (SUM and COUNT constraints). We detail each step below. The area set in Figure 3.1a is used as a running example to illustrate different steps of the construction. For simplicity and without loss of generality, we assume that all constraints are imposed on the same attribute s.

Step 1: Filtering and Seeding. The first step satisfies extrema constraints, i.e., MIN and MAX constraints. These two constraints have two functionalities: **(a) filtering out the invalid areas, and (b) specifying the set of seed areas**. Invalid areas are excluded during the feasibility phase (Section 3.5.1). We select seed areas as the areas that meet lower and upper bound values of one of MIN or MAX constraints. For instance, for two constraints $c_1 = (MIN, s_1, l_1, u_1)$ and $c_2 = (MAX, s_2, l_2, u_2)$, any area with $l_1 \leq s_1 \leq u_1$ or $l_2 \leq s_2 \leq u_2$ is a valid seed area. This rule is generalizable to any number of MIN and MAX constraints.

This step provides three levels of optimizations through choosing a seed area that satisfies the bounds of only one constraint. First, it finds seed areas for any arbitrary combination of MIN/MAX constraints. For constraints that are imposed on different attributes or have disjoint upper and lower bounds, there is no single area that satisfies all of them.

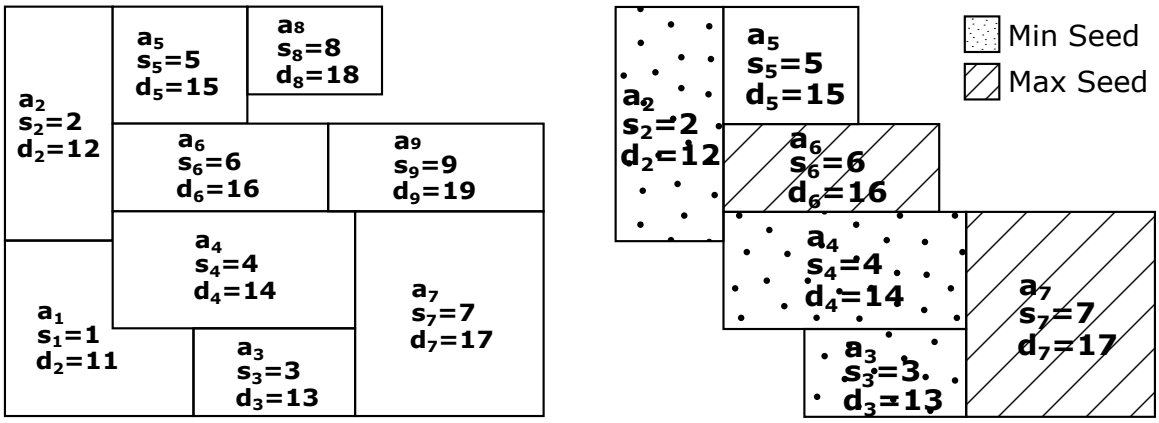
So, considering each constraint in isolation from others provides flexibility to find seeds for any query. Second, the number of seed areas is an upper bound for the number of regions p as each feasible region must contain at least one seed area for each constraint. So, using a maximum number of seed areas and growing the regions based on them contributes to the objective of maximizing p. Third, it enables piggybacking the seed areas selection on the invalid areas filtration during the *feasibility phase*. While checking the validity of an area, the algorithm checks the seeding conditions and mark seed areas.

Example: Figure 3.1b illustrates the result of Step 1 for the example area set in Figure 3.1a. The extrema constraints are set as $\{(MIN, s, 2, 4), (MAX, s, 6, 7)\}$. Areas a_1, a_8, and a_9 are moved to set U_0 because their attribute values are below 2 or above 7. The remaining areas after filtering are shown in Figure 3.1b. Area a_2, a_3, and a_4 are selected as seed areas for the MIN constraint and area a_6 and a_7 are selected as seed areas for the MAX constraint, as shown in Figure 3.1b.

Complexity analysis. All operations of the feasibility phase and Step 1 of the construction phase are performed in one pass over the n areas. Each area is validated against all MIN, MAX, and SUM constraints to be classified as either invalid, valid, or seed area. The time complexity is $O(m \times n)$, where m is the number of constraints and typically $m << n$.

Remark 1. *The time complexity of the feasibility phase and the Filtering and Seeding is* $O(m \times n)$.

Step 2: Region Growing. The second step grows regions that satisfy the AVG constraint $c = (AVG, s, l, u)$, without violating the MIN/MAX constraints. This step is



(a) Example area set (b) Area filtration and seed area selection

Figure 3.1: Step 1: Filtering and Seeding

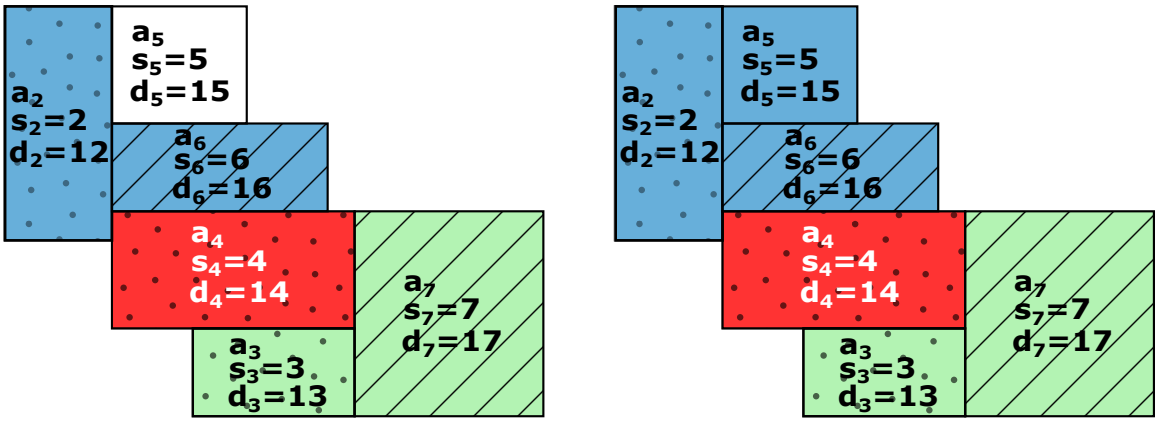
divided into three substeps. The first substep initializes a set of regions. The second substep assigns the unassigned areas to the regions. The third substep combines regions to ensure each region satisfies all constraints. We detail each substep below.

Substep 2.1 utilizes the set of seed areas, called *seeds*, from Step 1 to initialize a set of initial regions. Areas in *seeds* already satisfy at least one of the MIN/MAX constraints, so they must be included in any valid region. The algorithm iterates over *seeds* and classifies all seed areas into three subsets based on their AVG attribute value s: *unassigned_avg*, *unassigned_low*, and *unassigned_high*. *unassigned_avg* contains areas that satisfy the condition $l \leq s \leq u$. *unassigned_low* contains seed areas that satisfy $s < l$, i.e., s value is lower than c's lower bound. Similarly, *unassigned_high* contains seed areas with $s > u$. Only areas in *unassigned_avg* satisfy c and are used in region initialization directly. As we want to maximize the number of output regions p, we make each area in *unassigned_avg* a separate region, and all new regions are added to a region list P. Then, we merge areas in

unassigned_low and *unassigned_high* with their spatial neighbors to compose regions that satisfy c. Algorithm 1 gives the merging procedure.

Algorithm 1 initiates each unassigned area as a temporary region R (Lines 4 to 6). While R does not satisfy the constraint c's bounds, the algorithm tries to add a neighbor area a_n that moves R's overall average of attribute s towards c's range (Lines 16 to 21). Once R satisfies c, it is added to the region list P (Lines 10 to 12). If R's neighbors are exhausted and cannot form a valid region, the whole procedure is reverted, and areas of R remain unassigned.

Substep 2.2 tries to assign all remaining unassigned areas, either seed areas or regular areas. Similar to classifying *seeds* areas in Substep 2.1, all areas that are not in *seeds* set are added to either *unassigned_avg* set, *unassigned_low* set, or *unassigned_high* set, depending on their s value. There are **two rounds** for assigning the remaining unassigned areas. In the first round, we try to add the unassigned areas to their neighbor regions. All areas in *unassigned_avg* can be safely added to any of its neighbor regions as it is guaranteed not to introduce a violation of c. For each area in *unassigned_low* and *unassigned_high*, we must check if adding it to the neighbor region violates c. The second round tries to assign the areas in *unassigned_high* and *unassigned_low* by merging the regions. Given an unassigned area a, the algorithm tries to merge one of a's neighbor regions R and R's neighbor region with a and checks if c is satisfied for the newly merged region. The merging process terminates when all a's neighbors cannot absorb it or a merge limit, i.e., the number of allowed merge trials, is reached. The merge limit is set to prevent the formation of oversized regions and control the runtime.



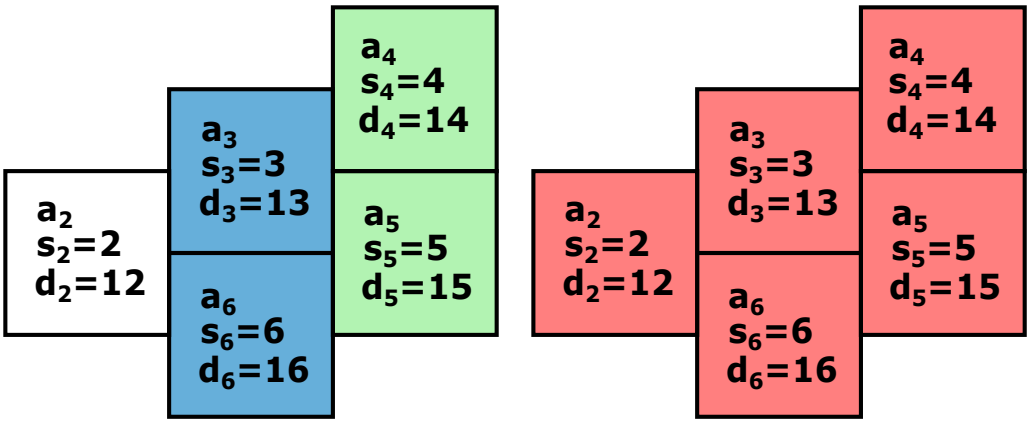
(a) Substep 2.1: Region Initialization

(b) Substep 2.2: Area Assignment

Figure 3.2: Step 2: Region Growing - Substep 2.1 & 2.2-Round 1

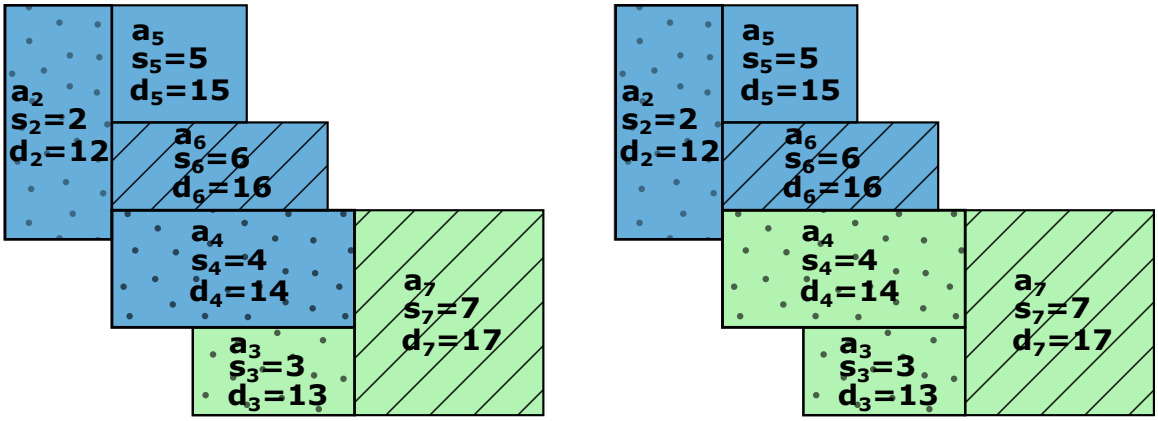
Substep 2.3 combines neighbor regions to ensure that all regions satisfy all MIN and MAX constraints. Because areas that form *seeds* are selected by a single MIN or MAX constraint, regions that are aggregated based on a single seed area are guaranteed to satisfy only one of those constraints. To ensure each region satisfies all constraints, the algorithm iterates over the region list P and merges the region that does not satisfy every constraint with one of its neighbor regions that satisfy other constraints. The iteration stops when all regions in the P satisfy MIN and MAX constraints, i.e., each region contains at least one seed area for each MIN/MAX constraint. This substep does not affect satisfying the AVG constraint c because merging two regions that already satisfy c produces a region that still satisfies it. Thus, after Step 2 concludes, all regions in the region list P satisfy all the MIN, MAX, and AVG constraints.

Example: Figure 3.2 shows an example of Substep 2.1 and Substep 2.2. Given $c = (AVG, s, 4, 5)$, in Substep 2.1, only seed area a_4 is added to *unassigned_avg* and initialized



(a) Unassigned area a_2. (b) Region merging and Assigning a_2.

Figure 3.3: Step 2: Region Growing - Substep 2.2 - Round 2



(a) Case 1 (b) Case 2

Figure 3.4: Step 2: Region Growing - Substep 2.3

as a new region R_{red} (Figure 3.2a). a_2 and a_3 are added to *unassigned_low* and a_6 and a_7 are added to *unassigned_high*. The average of $a_2.s_2$ and $a_6.s_6$ is 4, and they are combined to form the region R_{blue}. Similarly, a_3 and a_7 form the region R_{green} with average value 5. So, Substep 2.2 initializes three regions as depicted in Figure 3.2a. Then, in Substep 2.2, the remaining unassigned area a_5 is assigned to its neighbor region R_{blue} as depicted in Figure 3.2b.

Example: Figure 3.3 provides an example for Round 2 of Substep 2.2. In contrast to the example in Figure 3.2, the unassigned area a_2 cannot be added to its neighbor region $R_{blue} = \{a_3, a_6\}$ directly. Adding a_2 to R_{blue} generates a region with an average s value equals 3.67, which is smaller than 4 and violates the constraint $c = (AVG, s, 4, 5)$. Instead, the algorithm forms a temporary region $R_{red} = \{a_3, a_4, a_5, a_6\}$ by merging R_{blue} and its neighbor region $R_{green} = \{a_4, a_5\}$, as shown in Figure 3.3b. R_{red} accepts a_2 and the average value is 4.4, which satisfies the constraint c.

Example: Figure 3.4 gives an example of Substep 2.3 for the example region partition in Figure 3.2b. $R_{red} = \{a_4\}$ contains only the MIN seed area; hence it does not satisfy the MAX constraint. Because both neighbor regions $R_{blue} = \{a_2, a_5, a_6\}$ and $R_{green} = \{a_3, a_7\}$ satisfy the MAX constraint, R_{red} can be combined with either region to form a region satisfying all constraints. The results are depicted in Figure 3.4a and Figure 3.4b, respectively.

Complexity analysis. The Substep 2.1 and the first round of the Substep 2.2 iterate through all seed areas and each takes $O(n)$ time. The second round of the Substep 2.2 iterates through the unassigned area set for multiple iterations. In each iteration, at least one area is removed from the unassigned area set or otherwise no update is made and the loop breaks. The number of unassigned areas is $\leq n$, hence the worst-case total number of operations is $\sum_{i=1}^{n} i = O(n^2)$. The merge operations are bounded by a constant number and each operation takes constant time to update the attribute values and append the area list. Substep 2.3 iterates through all regions once, thus taking maximum of $O(n)$ time. Thus, Step 2 time complexity is $O(n) + O(n) + O(n^2) + O(n) = O(n^2)$.

Remark 2. *The time complexity of Region Growing is* $O(n^2)$.

Step 3: Monotonic Adjustments. Step 3 satisfies SUM and COUNT constraints while maintaining the constraints that have been satisfied in the previous steps. The SUM constraint is used in the MP-regions problem, hence we build this step based on the MP-regions construction algorithms [95]. However, because regions are initialized for the MIN, MAX, and AVG constraints, adjustments are needed on both area level and region level to satisfy the counting constraints. Because SUM and COUNT constraints are monotonic, areas are added to regions that fall under the lower bound or removed from regions that go above the upper bound to make every region valid. We iterate over constructed regions. For each region that violates one of the SUM or COUNT constraints, the algorithm first tries to swap areas with neighbor regions without violating the other constraints. The spatial contiguity of the region is validated by ensuring that the donor region still forms a single connected component after swapping. After no updates can be made by swapping areas, the algorithm tries to merge regions below the lower bounds or remove areas from regions above the upper bounds. When no changes can be made, the infeasible regions are removed, and the partition is considered finalized for the construction phase.

Example: With the input shown in Figure 3.4a and the example constraints $c_4 = (SUM, s, 12, \infty)$ and $c_5 = (COUNT, s, -\infty, 4)$, the region $R_{green} = \{a_3, a_7\}$ does not satisfy c_4. The boundary area with the neighbor region $R_{blue} = \{a_1, a_4, a_5, a_6\}$ is a_4. Area a_4 is swapped from R_{blue} to R_{green} after ensuring that both regions still satisfy all the other constraints, and R_{blue} is spatially continuous and still above the lower bound of c_4. The

result is depicted in Figure 3.4b. After swapping, the receiver region $R_{green} = \{a_4, a_5, a_7\}$ satisfies all the constraints, so the swapping attempts are terminated.

Complexity analysis. Each area is swapped at most once because the region that becomes feasible after the swap keeps the area to remain valid. Each swap updates and validates all attributes of both the donor and receiver regions, which takes $O(2m)$ operations. Checking donor region connectivity is $O(n)$ in the worst case. As a result, the time complexity of swapping is $O((2m+n) \times n) = O(n^2)$. The area removal of regions that exceed the upper bounds and removing infeasible regions take one $O(n)$ pass each. Thus, the overall time complexity is $O(n^2) + O(n) + O(n) = O(n^2)$.

Remark 3. *The time complexity of Monotonic Adjustments is* $O(n^2)$.

3.5.3 Local Search Phase

The partition with the largest number of regions p is forwarded from the construction phase to the local search phase to optimize the overall regions' heterogeneity. The local search phase uses the Tabu search algorithm [68], a meta-heuristic search algorithm, to minimize the overall heterogeneity.

In brief, the Tabu search algorithm keeps moving areas among neighbor regions without violating the user-defined constraints in any region. Starting from the feasible partition constructed in the construction phase, the Tabu search algorithm moves to the best neighboring partition. Moving to a solution that has worse total heterogeneity is allowed to escape from the local optimal solution. The made moves are stored in a tabu list, with a fixed tabu tenure, and the reverse moves are forbidden to prevent cycles. When

a move leads to a partition better than the best partition so far, the algorithm chooses this move even if the tabu list prohibits it. The algorithm stops when no better move can be made in a specified number of steps, and the best partition found is returned as the final result. This phase does not change the number of regions p. Instead, it still produces p regions but with better overall heterogeneity.

Complexity analysis. The Tabu search is an incomplete algorithm, so we derive a complexity approximation based on the parameters and the problem-specific neighborhood computation. Tabu search performs n iterations without heterogeneity improvement and the counter resets on improvement, so the total number of iterations is $O(\alpha n)$ in the worst case, where α is the number of moves that beat the existing optimal heterogeneity score. In each iteration, we update the valid moves using a region connectivity check for each boundary area in the region updated by the previous move, which takes $O(n \times n)$ time in the worst case. A move attempt takes $O(n)$ to compute the pair-wise dissimilarity in the worst case. So, the overall time complexity is $O(\alpha n \times (n^2 + n)) = O(\alpha n^3)$. Experimentally, the moves that improve heterogeneity happen at an early stage of the local search, causing the total number of iterations much smaller than $2n$.

Remark 4. *The time complexity of the Local Search phase is $O(\alpha n^3)$, where α is the number of valid moves that improve the heterogeneity over the existing optimum.*

3.5.4 Handling Arbitrary Sets of Constraints

Our discussion so far assumes the incoming query contains all types of constraints. However, it is common to have a subset of these constraints. For example, a query could

have only one AVG constraint, or one MIN constraint and one COUNT constraint. This section discusses how the $FaCT$ algorithm handles queries with an arbitrary subset of constraints.

The $FaCT$ algorithm handles absent constraints as they exist with an infinite range $(-\infty, \infty)$, which affects both the feasibility and the construction phases. If a MIN/MAX constraint type is absent, the feasibility phase does not remove any invalid areas under MIN/MAX, and Step 1 of the construction phase selects all areas as the seed areas. If the AVG constraint is missing, the Region Growing step adds all areas in *seeds* to *unassigned_avg* and then initializes single-area regions. Areas not in *seeds* are then added to *unassigned_avg* and merged to neighbor regions iteratively. If SUM/COUNT constraints are absent, Step 3 of the construction phase is omitted. Therefore, designing the $FaCT$ algorithm with independent steps enables it to effectively handle queries with any subset of constraints.

3.6 Complexity Analysis

This section analyzes the time and space complexity of the $FaCT$ algorithm. According to Remarks 1, 2, and 3 in Section 3.5.2, the time complexity of the feasibility and the three steps of the construction phases are $O(n)$, $O(n^2)$, and $O(n^2)$, respectively. Thus, the total time complexity of both phases is $O(n^2)$. According to Remark 4 in Section 3.5.3, the time complexity of the Local Search phase is $O(\alpha n^3)$. This gives the overall worst-case time complexity of $FaCT$ as $O(\alpha n^3)$, where α is the total number of moves that improves the existing optimal heterogeneity in the local search process.

For space complexity, the area set takes $O(m \times n)$ space to store the n areas, each associated with m attributes assuming there are m constraints. The space complexity for the seed area selection is $O(n)$. The Region Growing step stores temporary regions with $O(n)$ areas. After the region initialization, each region is stored with m attributes and an area set. The regions are disjoint, so the space taken by the region list is bounded by $O(m \times n)$. The Monotonic Adjustments step and the Local Search phase update the region list in place and does not require additional space. Thus, the overall space complexity is $O(m \times n)$.

3.7 Experiment evaluation

This section presents an extensive experimental evaluation of our proposed work to address the EMP problem. The code and data for reproducing the results presented in this section are publicly available [96]. The rest of the section introduces the experimental setup (Section 3.7.1), studies the impact of different constraints types and threshold ranges on the performance (Section 3.7.2), shows the scalability of the $FaCT$ algorithm (Section 3.7.3), and summarizes the experimental results (Section 3.7.4).

3.7.1 Experimental Setup

EMP is a novel problem in the literature, so there is no direct competitor to compare our techniques against. In addition, due to the major enrichment EMP introduces to the MP-regions problem, existing techniques that address MP-regions problem cannot be trivially extended and adopted as baselines to solve the EMP problem. Therefore, our

experiments focus on evaluating the performance and effectiveness of our proposed work. We also study the impact of different combinations of enriched constraints.

Evaluation datasets. We use nine real datasets that represent the census tracts of the USA, outlined as follows: (1) the Los Angeles City (denoted as LACity) that includes 1012 areas, (2) the Los Angeles County (denoted as LACounty) that includes 2344 areas, (3) Southern California (denoted as SCA) as identified by the Southern California Association of Governments (SCAG) [97] that includes 3947 areas, (4) the State of California (denoted as CA) that includes 8049 areas, and (5) Five multi-state datasets with ∼10k to ∼50k areas, as detailed in Table 3.1. Our default dataset is LACounty. Typical evaluation datasets in the existing literature have between 300-3000 areas [63, 95], so our default dataset size is comparable to the largest datasets in the literature. All datasets are joined with real attributes from 2010 US census data about facts in each census tract. All datasets share the same three spatial extensive attributes as shown in Table 3.2, one per constraint type, and the dissimilarity attribute is *HOUSEHOLDS*. *POP16UP* attribute represents the population with age sixteen or above. *EMPLOYED* attribute refers to the employed population. *TOTALPOP* is the total population. *HOUSEHOLDS* is the number of households in each area, and it is used to measure region heterogeneity.

The evaluation attributes are selected based on factors that influence the population growth rate [33], which makes the partitions obtained through the experiments useful for studying population growth. The shapefiles of the census tracts and the attribute tables are provided by the US Census Bureau [98] and SCAG [99] and joined using the QGIS software [100].

All experiments are based on Java 14 implementations using an Intel Xeon W-2123(3.60 GHz) and 20GB RAM allocated under Windows 10. We use three performance measures: the runtime for both construction and local search phases, the answer set size p, and improvement in regions' heterogeneity, defined as the ratio of the absolute difference between the heterogeneity score before and after the local search phase to the heterogeneity score before the local search. Our parameters include threshold range and dataset size. Unless mentioned otherwise, the default dataset is LACounty, area pickup criteria are random, the AVG merge limit is three, the length of the tabu list equals ten, and the maximum number of moves without improvement for Tabu search equals the dataset size. The default threshold ranges and attributes are shown in Table 3.2 for each constraint type. For space limitation and similarity of results on aggregates of the same type, we present results for one aggregate function in each constraint type.

3.7.2 Impact of Constraints Types

This section studies the impact of different sets of constraints on regionalization performance. Sets of constraints vary in terms of: (1) set size, i.e., number of constraints, (2) types of constraints, and (3) threshold ranges. Sections 3.7.2, 3.7.2 and 3.7.2 evaluate sets that include MIN, AVG, and SUM constraints, respectively. MIN constraints are denoted as *M*, combinations of MIN and AVG constraints are denoted as *MA*, combinations of MIN and SUM constraints are denoted as *MS*, combinations of MIN, AVG, and SUM constraints are denoted as *MAS*. Time of local search is denoted as *Tabu*.

MIN Constraint

This section evaluates regionalization queries with MIN constraints, $c = (MIN, s, l, u)$, in combination with other constraint types. MIN and MAX constraints perform two roles: filtering invalid areas and selecting seed areas, and both depend on the range threshold $[l, u]$. Hence, we explore three cases: (a) ranges with $l = -\infty$, which show the impact of u values on seed area selection, (b) ranges with $u = \infty$, which show the impact of l values on invalid area filtration, and (c) ranges with bounded values of l and u that show their simultaneous impact.

Ranges with $l = -\infty$. Figure 3.5 shows the runtime of the construction phase and Tabu search. The runtime highly depends on the types of constraints. Table 3.3 gives the number of regions p for different constraints combinations. In all cases, the single MIN constraint, M, produces the maximum of regions p that is bounded by the number of seed areas. The p value increases with the increase of the upper bound u. As u increases, the number of seed areas increases and it takes fewer iterations to partition the areas into regions, so the runtime decreases in Figure 3.5. When the other constraints are added, the regions in the final result contain more than one seed area, so p decreases. With more seed areas, the initialized region becomes smaller and the SUM constraint takes more operations to satisfy. This causes the runtime of MS and MAS to increase by a small amount although the time for MIN and AVG actually decreases. Increasing u leads to increasing the heterogeneity improvement from 6.96% at $u = 2k$ to 40.2% at $u = 5k$ due to higher p.

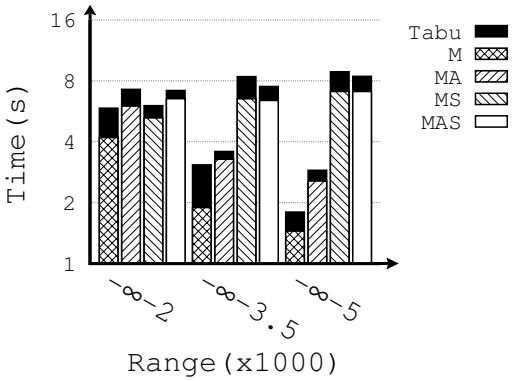


Figure 3.5: Runtime for MIN with $l = \infty$

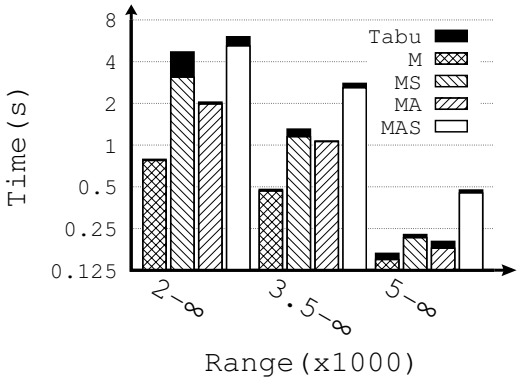
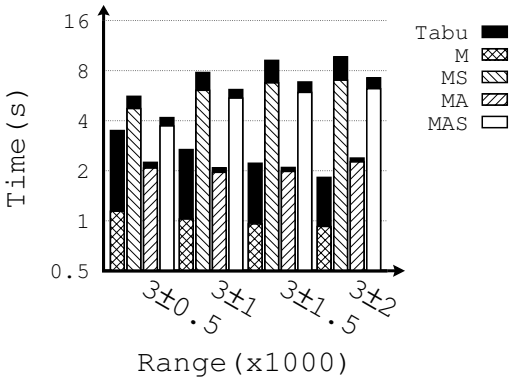
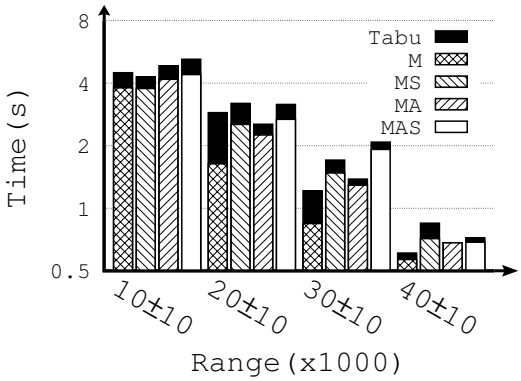


Figure 3.6: Runtime for MIN with $u = \infty$



(a) Varying range lengths



(b) Varying range midpoints

Figure 3.7: Runtime for MIN with bounded l and u

Ranges with $u = \infty$. Table 3.3 shows the p values and Figure 3.6 gives the construction time and local search time for different constraint combinations. As we increase the lower bound l, more invalid areas are filtered out, while typically the remaining areas are scattered. As a result, the p value decreases, and hence runtime decreases significantly due to the lower number of regions. The heterogeneity score improvement reduces as l increases with up to 17.6% due to decreasing p that narrows the search space of the local search.

Ranges with bounded l and u. We explore bounded ranges that vary the range length while fixing the range midpoint. Table 3.3 gives the p values and Figure 3.7a shows the runtime. When the range length increases, the p value increases because fewer areas are filtered out and more regions are initialized. As a result, both total and construction time increase due to larger search space. However, there is no clear trend for the Tabu search time.

For a range with a fixed length and shifted midpoint to a higher value, both construction time and local search time decrease, as shown in Figure 3.7b. For larger values, the area set is chopped into small connected components but the number of seed areas is relatively stable. As a result, Steps 2 and 3 of the construction phase terminate faster and reduce runtime. The p value depends on the correlation between attributes of MIN and AVG constraints. If they are not correlated, then the p value strictly decreases for larger midpoint values. When they are correlated, p might increase or decrease, depending on the correlation. In all cases, the heterogeneity improvement increases when p increases, by up to 11.3% in this case.

AVG Constraint

The average aggregate function is non-monotonic. Satisfying the constraint can be computationally heavy with certain ranges. To illustrate this, we explore two cases: (a) ranges with fixed length and changing midpoints, and (b) ranges with a fixed midpoint but varying lengths.

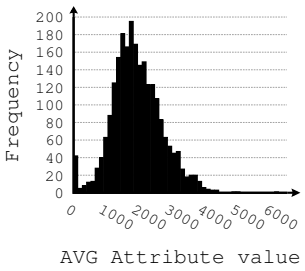
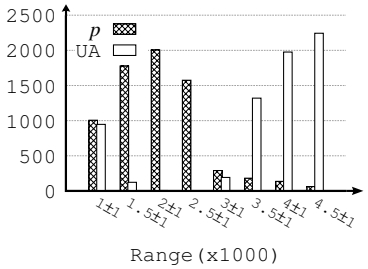
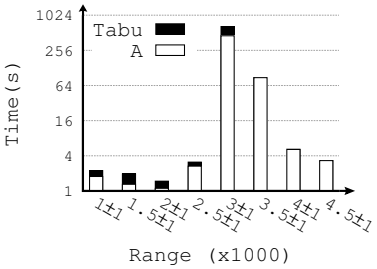


Figure 3.8: Distribution of AVG attribute value

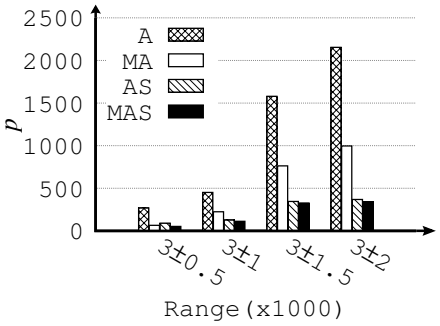


(a) p and UA

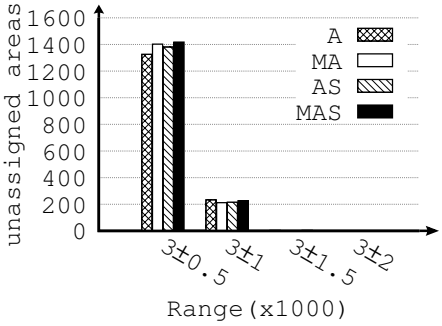


(b) Runtime

Figure 3.9: Impact different AVG range midpoints



(a) p



(b) Unassigned areas

Figure 3.10: Impact different AVG range lengths

Ranges with fixed range length. In this experiment, the length of the AVG range is fixed to be $2k$ and the midpoint shifts from $1k$ to $4.5k$, with the step size $= 0.5k$. We focus on the performance of the AVG constraint to exclude the impact of other constraints. Figure 3.8 shows the distribution of the AVG attribute value of the areas in the default dataset, which is a positively skewed distribution. The AVG attribute values of most of the areas are below $4k$, but there are several outliers with values up to 6149. Figure 3.8a shows the p value and the number of unassigned areas (UA) for each range while Figure 3.8b gives the runtime. When the midpoint is set to be $1k$ to $2.5k$, about half of the areas lie within the given range and it is also relatively easy to combine areas whose value is outside the range. As a result, the number of unassigned areas is reduced to 0 when the midpoint

increases to $2k$ and $2.5k$ but the runtime is below $3.2s$. The ranges are also flexible for the Tabu search phase, with the heterogeneity score improved by up to 30.1%. When the range is set to be $3k \pm 1k$, more than half of the areas lie below l but it is possible to combine them with one of the outliers whose attribute value is well above the upper bound. In addition, as an area is assigned, the newly formed region may lead to the possibility of assigning the neighbor areas. Hence the enclave assignment process continues for multiple iterations until no further update can be made. As a result, it takes a much longer time, but the percentage of unassigned areas is reduced significantly to 9%. When the midpoint is further increased to above $3.5k$, it is too hard for the algorithm to form feasible regions as almost all areas fall below the lower bound of the range. As a result, most areas remain unassigned and the construction time becomes much shorter as the algorithm quickly finds that no further changes can be made. When the midpoint is above $3k$, there are limited feasible moves for the Tabu search because the AVG constraint is too tight. As a result, the Tabu search time is negligible and the heterogeneity improvements are below 0.4%.

Ranges with fixed midpoint. Based on the observation from the previous experiment, we set the midpoint to be $3k$, which is the most challenging setting for the algorithm, and alter the length of the range. Combinations with the other constraints are also included to demonstrate how the AVG constraint can bottleneck the performance in special cases. The p values for different constraint combinations are given in Figure 3.9a and the runtime is shown in Figure 3.11. The figures show increasing p values with longer ranges, but the runtime depends on the range length. Different range lengths affect the runtime significantly, while different constraint combinations with the same range have

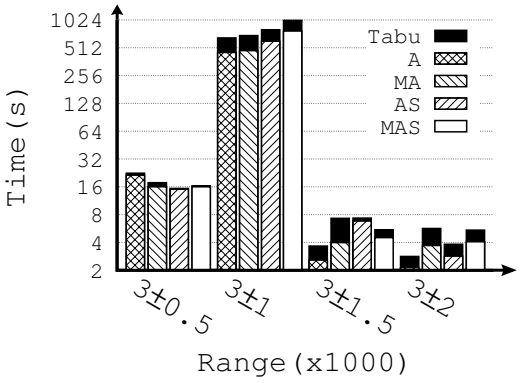


Figure 3.11: Runtime for AVG with different range lengths

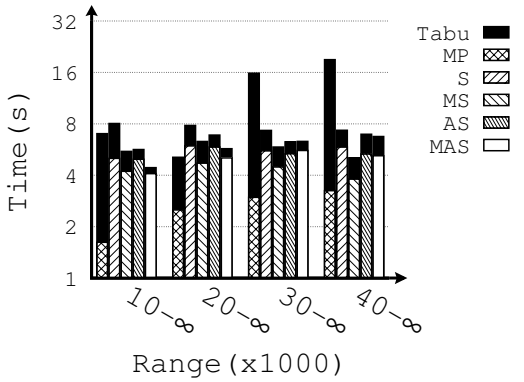


Figure 3.12: Runtime for SUM with $u = \infty$

much less impact on the construction time. This is because the runtime of the AVG step dominates the total runtime of the construction phase. When the range is $3k \pm 0.5k$, the range is so tight that the algorithm quickly finds no possible enclaves assignment, so the AVG step terminates early. However, 60% of the areas remain unassigned, as shown in Figure 3.9b. Similar to the previous experiment, the range $3k \pm 1k$ is the most challenging case and the runtime dominates the other constraints. However, the reduction in the number of unassigned areas is also significant for all constraint combinations. When the range is further enlarged, most of the areas lie within the range and it is much easier to assign the enclaves without merging. Thus, the time is much shorter, and all areas are assigned as well. The final heterogeneity score also reduces when the range is enlarged due to the reduction in region sizes. The heterogeneity improvement increases to up to 21.8% because when the range is enlarged, it becomes easier to find a valid move for the local search.

SUM Constraint

SUM constraints are the only type of constraints that are used in existing state-of-the-art solutions for the max-p regions (MP-regions) problem. We compare with them, denoted as MP, using a single SUM constraint with an open upper bound, i.e.,$[l, \infty)$. Table 3.4 shows the p values and Figures 3.12 and 3.13 give the runtime for different constraint combinations with SUM aggregate. At $u = \infty$, the p value decreases with increasing lower bound l while runtime does not increase as much for both MP-regions and $FaCT$. According to Table 3.4, our $FaCT$ algorithm gives comparable p value to MP-regions when the constraints are set to be the same. The construction time is slightly higher due to the overhead introduced by steps that validate the feasibility of the query and handle the other constraints. However, due to shorter tabu search time, the overall runtime of EMP is less than half of the runtime for the MP-regions when $u = 30k$ or $40k$. In addition, the $FaCT$ algorithm handles generic types of constraints, which are not supported by the competitor. The heterogeneity score increases when l increases, by up to 72.6% for both MP-regions and the single constraint S, 13.8% for MS, and less than 1% for AS and MAS due to the effect of AVG constraint. Increasing the region's size provides local search with more potential moves, hence it better improves overall heterogeneity. For bounded threshold ranges, in Table 3.4 and Figure 3.13, both p values and construction runtime decrease when the lower bound l increases and the trend is similar to $u = \infty$. The heterogeneity score increases by up to 30.4% for S, 6.9% for MS, 3.76% for AS, and 2.49% for MAS, with increasing range length due to the larger region size. However, the unassigned areas could reach up to 25.1% when u is bounded for MS, AS, and MAS. This is because areas are removed so

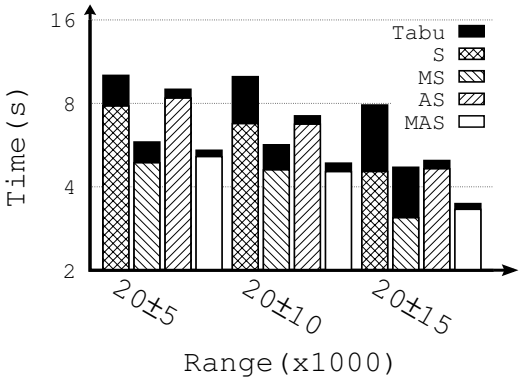


Figure 3.13: Runtime for SUM with a changing range length

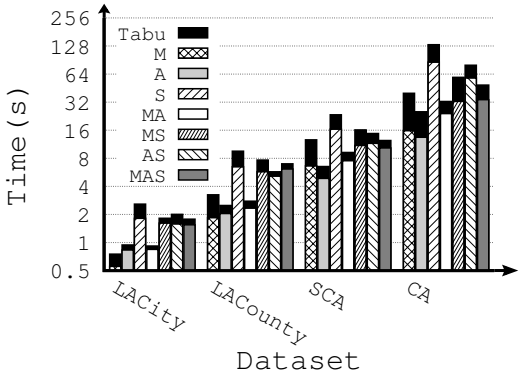


Figure 3.14: Runtime varying datasets LAcity to CA

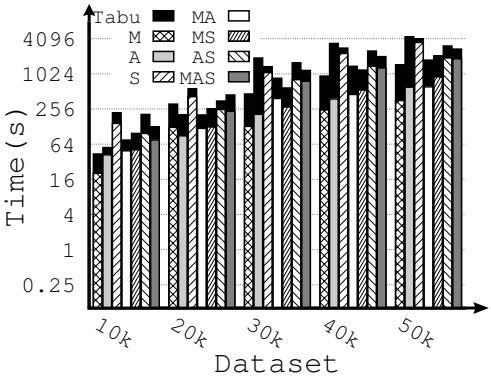


Figure 3.15: Runtime varying datasets for 10k-50k

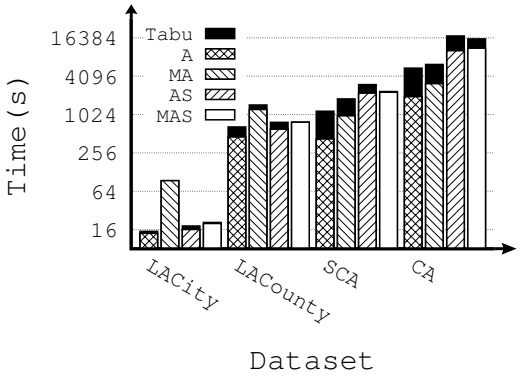


Figure 3.16: Runtime varying datasets for AVG constraint with range $3k \pm 1k$

that each region does not exceed u. *FaCT* algorithm reports output statistics to users so they are equipped with information about the impact of different threshold ranges on the given dataset, and are able to tune query parameters insightfully.

3.7.3 FaCT Scalability

This section discusses the scalability of the *FaCT* algorithm. We run the algorithm for different constraint combinations with the default threshold values on the nine evaluation

datasets. The dataset size of the 50k dataset is 17 times the largest dataset used in the existing literature of regionalization [63, 95]. As the AVG constraint is identified as a clear performance bottleneck when the range is set to be $3k \pm 1k$, our discussion contains two parts. The first part shows the results for the constraint combinations with the AVG set to be default range (Figure 3.14 and Figure 3.15) to show the scalability of $FaCT$ in normal cases, while in the second part we focus on constraint combinations including AVG with range $3k \pm 1k$ (Figure 3.16) to see how the computation of the AVG constraint scales in extreme cases.

Figure 3.14 and 3.15 show that when the AVG constraint is not the bottleneck, the runtime increases linearly with the input size for M and quadratically for the other constraints. For all datasets, $FaCT$ provides a very acceptable runtime for regionalization applications.

Figure 3.16 shows that when the range of the AVG is set to be $3k \pm 1k$, AVG constraints increase the construction time to a large extent.. For other datasets, the runtime increases with increasing input size at a much higher rate than the previous experiment, which implies less scalability advantage for AVG constraint. The construction runtime does not strictly increase with increasing the input size. The construction time for the SCA dataset is shorter than that for the LACounty dataset, except for AS and MAS. This difference is caused by the merging procedure in the AVG constraint. Generally, more areas are easier to aggregate in regions that satisfy the AVG constraint. In all cases, the construction time scales much better than the Tabu time, and it still provides a solution with almost the same quality.

3.7.4 Summary Of Results

All experiments show that the *FaCT* algorithm can efficiently produce feasible solutions of high quality in a few seconds for the vast majority of the cases. In addition, the *FaCT* algorithm supports relatively large datasets that serve all existing applications. When more areas are filtered or fewer seed areas are identified by the extrema constraints, both p and the runtime tend to decrease. The behavior of the centrality constraint depends on the range specified and the distribution of the attributes. However, it shows reasonable performance in the vast majority of the cases. The counting constraints can formulate the MP-regions problem as a sub-problem of EMP and gives comparable scalability and result quality.

3.8 Conclusion

This chapter introduces an enriched max-p (EMP) regionalization problem that extends the existing regionalization problems with SQL-inspired user-defined constraints. The EMP problem clusters a set of spatial areas into homogeneous regions that satisfy the user-defined constraints. The EMP problem enables enriched types of constraints that support SQL-inspired aggregate functions, MIN, MAX, AVG, SUM, and COUNT, with range operators. We prove the NP-hardness of the EMP problem. To tackle this problem, we propose *FaCT*; a three-phase algorithm that finds an approximate solution with a maximum number of regions and minimum overall heterogeneity. The first phase checks the feasibility of finding a solution given the input constraints. It also provides users with insightful information to tune their input and enable flexible exploration for various datasets.

The second phase constructs an initial solution, and the third phase further optimizes it to provide a final solution. We have empirically demonstrated the capability of $FaCT$ to provide efficient and effective performance on various real datasets.

For future work, we explore multi-objective optimization methods and further improve the algorithm performance through parallelization.

3.9 Extension

Based on the EMP framework, we made two major extesnsion: the support for the variance extension and the incremental update.

3.9.1 Varaince constraint

As one of the core statistical aggregate functions, variance measures the dispersion of the data and serves as an important reference in multiple fields. In regionaliztion applications, variance aggregate can be adopted to define diverse or homogenious regions. For example, by specifying a large variance target for natural resource distributions accross different areas with in a region, researchers can obtain regions with various soil types, temperatures, and rain fall distributions which can potentially accommodate a great variety of species. In addition, variance in population with different cultural background also suggests a more diverse society. More important, with the range operator that EMP supports, researchers will be able to guide the tendency of the region in termes of heterogeneity to observe more meaningful patterns.

Computation: The computation of the VAR constraint closely parallels the three-step procedure of the AVG constraint. Both constraints are nonmonotonic, but they exhibit a key distinction. The VAR constraint specifically manages the dispersion of regions. Consequently, it's challenging to directly determine if a region remains within the defined range after the addition of an area or merging without particular computations. To address this, we introduce two equations that enable efficient updates to a region's variance attribute and provide insights into the viability of area assignments or region mergers.

For each region, we continuously update two sums: the sum of variance attribute values $\sum s$, and the sum of the squares of these values $\sum s^2$. The symbol $\sum$ here aggregates the values for all areas within a given region. The region's variance can be computed as:

$$\begin{aligned} VAR(R_i) &= \frac{\sum(s_j - \bar{s})^2}{|R_i|} \\ &= \frac{\sum(s_j^2 - 2\bar{s}s_j + \bar{s}^2)}{|R_i|} \\ &= \frac{\sum s^2}{|R_i|} - \frac{2\bar{s}\sum s}{|R_i|} + \frac{|R_i|\bar{s}^2}{|R_i|} \\ &= \frac{\sum s^2}{|R_i|} - 2\bar{s}^2 + \bar{s}^2 \\ &= \frac{\sum s^2}{|R_i|} - (\frac{\sum s}{|R_i|})^2 \end{aligned} \tag{3.7}$$

By leveraging Equation 3.7, we can deduce the potential change in a region's variance attribute value upon the addition of an area by simply adjusting $\sum s$ and $\sum s^2$. Furthermore, this allows us to identify the permissible range of variance attribute values for areas that can be added to the region. If x symbolizes the variance attribute value of a potential area, the variance post its addition is represented as $VAR(R_i')$, where $|R_i'|= |R_i|+1$.

$$\begin{aligned} VAR(R_i') &= \frac{\sum s^2 + x^2}{|R_i|+1} - (\frac{\sum s + x}{|R_i|+1})^2 \\ &= \frac{\sum s^2 + x^2 - \frac{(\sum s + x)^2}{|R_i|+1}}{|R_i|+1} \\ &= \frac{\frac{|R_i|x^2}{|R_i|+1} - \frac{2\sum sx}{|R_i|+1} + \sum s^2 - \frac{(\sum s)^2}{|R_i|+1}}{|R_i|+1} \end{aligned} \tag{3.8}$$

To make sure the new region satisfies the VAR constraint, we have:

$$l \leq VAR(R_i') \leq u \tag{3.9}$$

Using Equation 3.9, we can swiftly determine and store the acceptable variance attribute value range for each region by finding the roots of the quadratic equations $VAR(R_i'|= l$ and $VAR(R_i') = u$.

We then delve into three substeps for area verification and assignment: Substep 2.1V utilizes the seed areas, named *seeds*, from Step 1 to initialize potential regions. Every area within *seeds* complies with at least one MIN/MAX constraint and must feature in any legitimate region. If $l \leq 0$, every seed initializes as a separate region, as any solo area inherently has a variance of 0. For scenarios with $l > 0$, we iterate across *seeds*, amalgamating seed areas with their neighbors to align the variance of the emerging region closer to the range $[l, r]$. During this addition, seed areas that satisfy a given constraint that the current temporary region does not are given preference. In all other instances, non-seed areas are prioritized to maintain the number of potential regions. The algorithm persists in adding neighboring areas until the temporary R meets the bounds of constraint c. Following this, R is added to the region list P. Should the neighbors of R be insufficient to form a valid region, the entire operation is reversed, leaving the areas of R unassigned.

Substeps 2.2V and 2.3V mirror the procedures of Substeps 2.2 and 2.3 for the average constraint respectively. In Substep 2.2V, the algorithm iterates through areas yet to be assigned, incorporating them into neighboring regions if their VAR attribute values align with the neighboring region's acceptable range. By Substep 2.3V, any region devoid

of a seed area for every constraint is combined with its adjacent regions. However, because merging two regions satisfying the VAR constraint doesn't ensure a resultant region staying within the range, owing to differing averages of the variance attributes, the merge limit is adopted here to allow the algorithm to attempt combining several regions before getting into the feasible range according to Equation 3.7. A similar heuristic mechanism is also adopted for the following step when an area reallocation or region merging violates the VAR constraint. In instances where unfeasible regions that can't merge with their neighbors persist, they are removed, with their areas incrementally integrated into neighboring regions as dictated by the acceptable range derived from Equation 3.9.

Runtime: The approach to compute AVG and VAR shares procedural similarities, and any single query can accommodate only one constraint at a time. As we have shown that the validation operations that are different for the two constraints all take constant time, the time complexity of the two constraints is the same.

3.9.2 Incremental update and reuse of results

Incremental update is a useful technique widely adopted in database systems and real-time data processing. Adapting the $FaCT$ algorithm to support the incremental update feature boosts the scalability when dealing with large datasets or bottlenecking constraints and also enables interactive exploration.

As stated in Section 6.1, one of the major concerns of the regionalization problems is how to define the scale of the region, which inspires the evolution from the p-regions problem to MaxP. With more complex constraints introduced to the EMP formulation, adjusting the scale takes a much longer time as the problem has become much more difficult.

In addition, as the size of the dataset increases, even the efficient operations, such as filtering and seeding that take $O(n)$ time, can be expensive. However, due to the greedy approach that $FaCT$ adopts and the selection scheme implemented, the results obtained for each constraint remain high quality as long as the corresponding constraint does not change.

We classify the changes into two categories and adopt different approaches to perform the incremental update operation accordingly:

Local Incremental Update: This pertains to alterations affecting a small portion of the regions. Historical changes in administrative boundaries, like merging or splitting counties and cities due to political or functional reasons, are typical examples. In terms of the EMP formulation, there can be a dramatic change in the population of certain areas, which will render the current partitioning obsolete. However, the changes are usually kept local, i.e. the merging and splitting of regions as well as the reallocation of areas happens within the neighboring counties or cities. With this observation and the inspiration from the Iterated greedy algorithm [101–103], we propose to perform a deconstruction and reconstruction to repartition the areas within the one-hop neighbor if the changes of attributes for the areas happen locally. To be more specific, the incremental update for local changes is divided into three steps. 1. We determine whether the regions that are affected by the changes are in a local cluster of one-hop neighbors, i.e. if a set containing a certain region and its neighbor regions contains all regions that are affected. 2. The one-hop neighbor is deconstructed and the areas are released while the other parts of the partition remain untouched. 3. We perform a reconstruction with the released areas and combine the updated partial result with the unchanged part from the previous result. As demonstrated in

Section 3.6, the time complexity of the *FaCT* algorithm stands at $O(\alpha n^3)$. Consequently, *FaCT*'s performance is markedly enhanced when the problem size is reduced by the local incremental update.

Global Incremental Update: This relates to significant alterations to one or multiple constraints. Such changes might involve users altering threshold values or switching to different attributes for a particular constraint, which affects all regions. Because of the modular design of the construction phase, the results of each step can potentially be reused when there is a partial update of the constraints or the attribute values are updated. For example, after the result is obtained for the constraints $C = \{(MIN, s_{min}, l_{min}, u_{min}), (AVG, s_{avg}, l_{avg}, u_{avg}), (SUM, s_{sum}, l_{sum}, u_{sum})\}$, a user might want to adjust the threshold for SUM to change the scale of the regions, or change s_{sum} from total population to employed population. In this case, the results obtained by Step 2 of the construction phase can be reused. Because the AVG computation can seriously bottleneck the performance, the ability to perform the incremental update saves a huge amount of resources and time and is especially in an online environment. We modify the *FaCT* algorithm to support the global incremental update in the following way: 1. We determine the constraints that are redefined by comparing the hash of the constraint definition and the attribute list. 2. The results that can be reused are determined by the unchanged constraints. 3. The partition is incrementally updated by only recomputing the required constraints.

Algorithm 1: Region Growing - Merging Areas

Input: $c = (AVG, s, l, u)$, *unassigned_low*, *unassigned_high*, P
Output: *unassigned_low*, *unassigned_high*, P

```
1  removed_areas ← ∅
2  u_areas ← unassigned_low ∪ unassigned_high
3  foreach area a ∈ u_areas do
4      Create a tempory region R with R.id = -1;
5      R.addArea(a);
6      updated = true;
7      neighbor_areas = R.getNeighbors();
8      while updated do
9          updated = false;
10         if l ≤ R.getAverage() ≤ u then
11             R.updateId(P.size() + 1);
12             P.add(R);
13             removed_areas.addAll(R.areas);
14         end
15         else
16             foreach area a_n in neighbor_areas do
17                 if a_n is not assigned to any region then
18                     if (R.getAverage() < l AND a_n.s > u) OR ((R.getAverage() > u
                         AND a_n.s < l)) then
19                         R.addArea(a_n);
20                         updated = true;
21                         break;
22                     end
23                 end
24             end
25         end
26     end
27 end
28 unassigned_low.removeAll(removed_areas);
29 unassigned_high.removeAll(removed_areas);
30 return unassigned_low, unassigned_high, P;
```

Name	No. of areas	States
10k	10255	CA, NV, AZ
20k	20570	10k + OR, WA, ID, UT, MT, WY, CO, NM, OK, NE, SD, ND
30k	29887	20k +TX, LA, AR, MO, IA
40k	40214	30k + MN, MS, AL, TN, KY, IL, WI
50k	49943	40k + GA, IN, MI, OH, WV

Table 3.1: Description of the multi-state datasets

Constraint Type	**Aggregate**	**Attribute**	**Range**
Extrema	MIN	POP16UP	$(-\infty, 3000]$
Centrality	AVG	EMPLOYED	[1500,3500]
Counting	SUM	TOTALPOP	$[20000, \infty)$

Table 3.2: Default attribute and ranges for different constraints.

	Ranges with l = -∞			Ranges with u = ∞			Ranges with bounded u and l							
	(-∞,2k]	(-∞,3.5k]	(-∞,5k]	[2k,∞)	[3.5k,∞)	[5k,∞)	[2.5k,3.5k]	[2k,4k]	[1.5k,4.5k]	[1k,5k]	[1k,2k]	[2k,3k]	[3k,4k]	[4k,5k]
M	270	1447	2172	2074	898	173	834	1501	1895	2109	207	789	712	401
MS	184	354	365	342	142	12	281	334	356	362	159	307	215	78
MA	208	1037	1483	1593	812	97	776	1206	1398	1496	175	654	704	386
MAS	170	335	341	337	145	7	275	328	339	344	152	304	215	74

Table 3.3: p values for different threshold ranges for MIN constraint combinations.

	Ranges with u = ∞					Ranges with different lengths		
	[1k,∞)	[10k,∞)	[20k,∞)	[30k,∞)	[40k,∞)	[15k, 25k]	[10k,30k]	[5k,35k]
MP	2298	717	373	245	185	N/A	N/A	N/A
S	2298	720	371	245	186	489	714	1358
MS	1056	581	342	237	179	408	567	785
AS	1545	642	345	233	177	445	640	1095
MAS	758	518	323	226	172	370	497	631

Table 3.4: p values for different threshold ranges for SUM constraint combinations.

Chapter 4

Scalable Evaluation of Local K-Function for Radius-Accurate Hotspot Detection in Spatial Networks

4.1 Introduction

In recent years, the rapid growth of spatial data has sparked considerable interest in the field of spatial analysis [104–106]. Spatial data is usually associated with spatial networks [107–113]. Spatial networks significantly influence various aspects of society, from daily commuting patterns [114] to the distribution of goods and services [115]. Hotspot detection along the spatial network is an important problem that identifies regions where

objects are significantly concentrated. Compared to Euclidean space [106], hotspot detection proves to be more precise along spatial networks, given the fact that most human activities are centered around these networks [105]. Hotspot detection in spatial networks is applied in diverse fields such as traffic management [110], epidemiology control [111], and crime analysis [107]. In some cases, practitioners having specific domain knowledge aim to identify hotspots within predetermined radii. For example, in environmental studies [116], certain pollutants tend to affect a specific radius around the point of release, so, fixed-radius hotspots could be established at this distance to pinpoint high-risk areas. However, in many cases, the hotspot radius may not be known, e.g., identifying crime regions and traffic congestion. Therefore, most hotspot detection algorithms used in spatial networks automatically determine the appropriate radius through enumeration. [106, 108, 117–119].

Existing hotspot detection methods along spatial networks fall into two categories: clustering-based methods [112, 113, 120–122] and statistical-based methods [104, 105, 108, 117–119, 123]. While clustering methods have efficient runtime, they might produce false-positive results [106]. In contrast, statistical-based methods offer a thorough statistical examination of the detected hotspots, typically by using a scoring function for quantification, e.g., log-likelihood score [108], density score [117, 118], to filter candidates. However, the log-likelihood score doesn't directly give a statistical significance guarantee because it doesn't take into account the variability that could arise from random sampling. Consequently, randomization techniques such as Monte Carlo Simulation [124] are followed to further validate the statistical significance. However, these kinds of approaches often demand a large number of simulations, which can lead to performance issues.

The network local K-function [105, 125] is a statistical function for the analysis of the distribution of objects in the spatial network. For a given center, it computes the sum of edge weights and the number of reachable objects in the network within a specific distance from the center and derives a distribution function. Consequently, the network local K-function can be used directly to compute a measure of statistical significance due to its incorporation of the distribution function, without using randomization techniques for statistical validation. This makes its computation efficient and solution quality statistically supported. Consequently, the network local K-function can efficiently and effectively detect hotspots in spatial networks.

Detecting hotspots in spatial networks is challenging for several reasons. Firstly, modern network datasets are extensive, including up to millions of objects [126], whereas existing statistical methods for network hotspot detection are typically designed to handle smaller datasets, usually thousands of objects [108, 117, 118]. Second, the distances between objects are constrained by the network topology, where edges in the network are intricate and interconnected, making the distance computation more complicated compared to that in Euclidean space. Third, determining the range of each hotspot is computationally expensive. Existing statistical-based methods for hotspot detection are mainly based on enumeration [108, 117, 118, 127], which does not scale due to the vast candidate size involved.

In this chapter, we address the limitations of the current hotspot detection algorithms in spatial networks based on the network local K-function. The algorithm is now

available in the Machine Learning module of the *Pyneapple* library. Our contribution can be summarized as follows:

- We propose two problems on spatial networks: Hotspot Detection with Predefined Radius (HDPR) and Hotspot Detection Without Predefined Radius (HDWPR) based on the network local K-function.
- We propose efficient algorithms to solve the HDPR problem.
- We define optimal hotspots radii and propose an approximate algorithm to solve the HDWPR problem.
- We present complexity analysis and correctness proof for the proposed algorithms.
- We conduct extensive experimental evaluations on both real and synthetic spatial network datasets to demonstrate the efficiency of our methods.

The rest of the chapter is organized as follows: Section 4.2 presents the related work. Section 4.3 defines the problems. Section 4.4 and Section 4.5 detail our proposed algorithms. Section 4.6 presents the experimental evaluation and Section 4.7 concludes the chapter.

4.2 Related Work

In this section, we discuss the work related to hotspot detection in spatial networks and the evaluation of the network K-function.

Hotspot detection: Spatial hotspot detection is an important problem with numerous applications, e.g., identifying regions with high occurrences of diseases in the

study of epidemiology [128,129], pinpointing locations with a high concentration of accidents or congestion in traffic management [110,130], and revealing areas with high crime rates in public safety [131, 132]. Hotspot detection has been studied extensively in the Euclidean space [106]. However, in this chapter we focus on hotspot detection in spatial networks as it provides accurate insights based on actual edge distances.

Consider a set of events happening on a spatial network, a *hotspot* [108] in a spatial network denotes a region in the network that exhibits a high density of events compared to the mean number of events. There are two main approaches for identifying hotspots in spatial networks: (1) *clustering-based hotspot detection*, and (2) *statistical analysis-based hotspot detection*. Clustering-based hotspot detection [112, 113, 120–122] aims to identify groups of events that are close to each other in the network space. Extensions of traditional clustering approaches have been developed to support spatial networks, e.g., partition-based clustering [121,133], DBSCAN-based [112,121,134], and hierarchical-based clustering [121]. While clustering-based hotspot detection algorithms are considered computationally efficient, they lack strong statistical robustness, i.e., they can result in biased or misleading results [106]. For example, partition-based clustering [121] and DBSCAN-based clustering [112] algorithms will always return clusters even if the points are randomly distributed. These clusters are not statistically robust and are considered false positives for statistical methods.

Statistical-based hotspot detection [104, 105, 108, 117–119] defines hotspots based on statistical models. These models identify regions where the density of events significantly exceeds the expected density assuming a specific event distribution. Khalid et al. [107] use

the NetKDE [130] to detect crime hotspots and the Getis-Ord GI* [135] is adopted as the statistical evidence of hotspot properties. However, this method is rather computationally expensive for large networks as it needs to compute the pairwise correlation between events. Tang et al. [108], propose NPP, which is the most relevant statistical method for objected-centered hotspot detection in spatial networks. NPP assigns a log-likelihood score for all possible radii of hotspots, and uses network partitioning to prune spatial objects that do not contribute to hotspots. Candidate hotspots are validated using Monte Carlo simulation trials [124]. While NPP provides strong statistical evidence for detected hotspots, NPP is computationally expensive because of the large number of Monte Carlo simulation trails and the scoring of multiple radii per hotspot.

The **network K-function:** [105] is proposed as a robust measure for density and distribution of spatial patterns in the network space. It is divided into two categories: the *network global K-function* [105, 109, 136] and the *network local K-function* [105, 125]. Chan et al. [109] introduced the state-of-the-art algorithm for the evaluation of the network global K-function, which counts the object pairs within a specific distance threshold. It speeds up the processing by sharing the processing of nearby objects, i.e., *neighborhood sharing*. One important limitation of the network global K-function is its inability to identify objects that are part of dense clusters.

Recently, the network local K-function [105, 125] has been introduced to better analyze the densities and distribution of spatial objects surrounding a center object. It calculates the total number of spatial objects that are within a specific network distance from the center object as well as the sum of network distances between those spatial objects

and the center object. The main advantage of the network local K-function is that it provides strong statistical evidence on the densities of spatial objects in the network.

In this chapter, we address the problem of efficient hotpot identification with strong statistical evidence using the network local K-function. The statistical evidence of the network local K-function is based on the spatial distribution function and density of data which provides strong statistical evidence of the spatial density with low computational overhead. Notice that, existing statistical-based hotspot detection algorithms [108,117,118] enumerate and score a large number of hotspot radii. This is computationally expensive, even with the proposal of pruning algorithms [108] that reduce the search space of hotspot radii. On the contrary, we identify hotspots with and without a predefined radius while avoiding the expensive enumeration. This significantly improves the efficiency of our algorithm without affecting its utility.

4.3 Preliminaries and Problem Definition

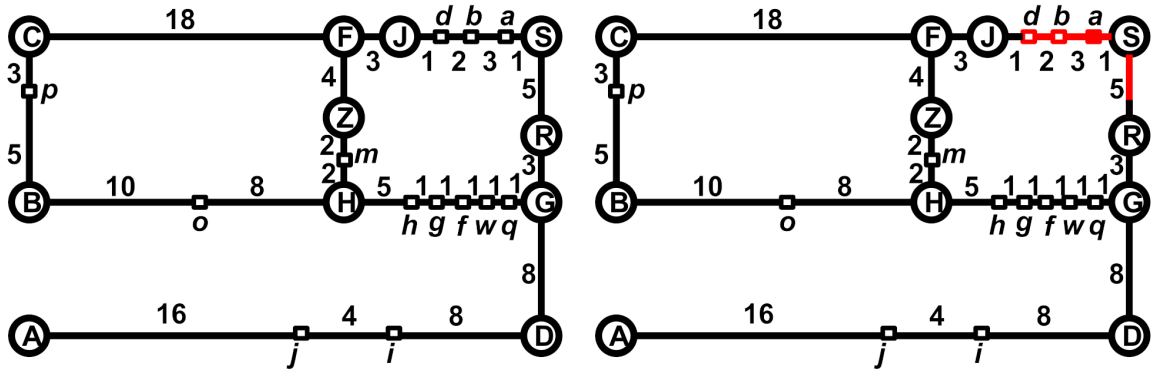
In this section, we formally define the terminologies used and the problems addressed in the chapter.

A ***spatial network*** [105], denoted as $G = (V, E)$, is an undirected weighted graph where V is a set of vertices and E is a set of edges. V has two types of vertices: (1) ***location vertices*** that represent physical locations, e.g., road intersections and turning points, and (2) ***event vertices*** that represent spatial events, e.g., vehicle locations, 911 calls, or traffic collisions. Practically, the set V is the union of two sets: a set of location vertices and a set of event vertices. The spatial events are overlaid over the physical straight lines that connect

physical locations. E represents the set of connections among graph vertices. $w(u, v)$ represents the weight of the connection (u, v) and it is the network travel distance between vertices u and v. Every connection $e \in E$ is either: (1) an ***edge***, or (2) a ***segment***. An edge is a direct connection between two location vertices. A segment is a direct connection that connects an event vertex to another event vertex or to a location vertex. Each segment is overlaid on an edge and each edge is associated with zero or more overlaid segments. Figure 4.1a gives an example of a spatial network where location vertices and event vertices are denoted by uppercase and lowercase letters, respectively. In Figure 4.1a, Edge (F, J) is an edge associated with zero segments, while Edge (J, S) is an edge associated with four segments (J, d), (d, b), (b, a), and (a, S). E maintains both edges and their overlaid segments as graph connections in one set differentiated with a flag attribute indicating their type. We denote the shortest path distance between any pair of vertices x and y, as $dist(x, y)$, where x and y can be location or event vertices. In Figure 4.1a, $w(J, S) = 7$, and event vertex o is overlaid on Edge (B, H) with $dist(o, H) = 8$, $dist(o, B) = 10$, and $dist(o, m) = 10$.

The number of overlaid event vertices along an Edge e in G is denoted as $|e|$. The number of location vertices and event vertices in G are denoted as $|V_l|$ and $|V_e|$, respectively, so $|V| = |V_l| + |V_e|$. The number of edges and segments are denoted as $|E_e|$ and $|E_s|$, respectively, so $|E| = |E_e| + |E_s|$.

An ***isodistance sub-network*** [108] $I(t|\vartheta)$ of an event Vertex ϑ with a given distance t is a network that consists of vertices, edges, and segments that are reachable from ϑ within a distance t. Edges and segments could be fully or partially covered by $I(t|\vartheta)$.



(a) Spatial Network (b) Isodistance sub-network

Figure 4.1: Spatial Network Example

$L(t|\vartheta)$ is the sum of weights of all the edges and segments in $I(t|\vartheta)$ and $L(G)$ is the total weight of edges in G. If an edge or a segment is partially covered by $I(t|\vartheta)$, only the covered parts are considered in $L(t|\vartheta)$. The weights of edges and their overlaid segments are not added twice. The number of event vertices in $I(t|\vartheta)$ is denoted as $N(t|\vartheta)$. Figure 4.1b shows an example of $I(5|a)$ where all the edges, segments, and the event vertices encountered in $L(5|a)$ are highlighted in red. In this example, $L(5|a) = 10$ and $N(5|a) = 3$ (counting event vertices a, b, d). In Figure 4.1a, $L(G) = 116$. Whenever there is no ambiguity, $L(t|\vartheta)$ is abbreviated as L and $N(t|\vartheta)$ is abbreviated as N.

The ***network local K-function*** [105, 125] is a statistical function to analyze the density and distribution of events surrounding a center vertex. This function takes two inputs, an event Vertex ϑ and a network distance t. Formally,

$$K(t|\vartheta) = \frac{N(t|\vartheta)}{\rho}$$

where $\rho = \frac{|V_e|-1}{L(G)}$ represents the overall density of event vertices within the entire network. In Figure 4.1a, $\rho = \frac{13-1}{116} = 0.103$, and $K(10|a) = \frac{3}{0.103} = 29$.

Complete spatial randomness (CSR) hypothesis [105] is a fundamental spatial statistical hypothesis that describes random occurrence of spatial events. CSR states that random locations of spatial events follow a homogeneous binomial point process [105]. We use the CSR hypothesis to represent a hypothetical distribution of spatial events surrounding a center vertex without hotspots. The expectation μ and variance σ^2 of the hypothetical spatial distribution surrounding ϑ are calculated as follows:

$$\mu(K(t|\vartheta)) = L(t|\vartheta)$$

$$\sigma^2(K(t|\vartheta)) = \frac{1}{|V_e|-1} * L(G) * L(t|\vartheta) * (1 - \frac{L(t|\vartheta)}{L(G)})$$

In Figure 4.1b, $\mu(K(5|a)) = L(5|a) = 10$, and $\sigma^2(K(10|a)) = \frac{1}{13-1} * 116 * 10 * (1 - \frac{10}{116}) = 88.33$.

For a specific radius t, an event Vertex ϑ is a **hotspot** when the actual density of events within t distance from ϑ exceeds the expected density. We find the quantile of the actual local K-function $K(t|\vartheta)$ compared to the hypothetical distribution. If it is above a preset threshold, e.g., 90%, ϑ is considered a hotspot [105]. Formally, ϑ is a hotspot when

$$K(t|\vartheta) > K_\alpha(t|\vartheta)$$

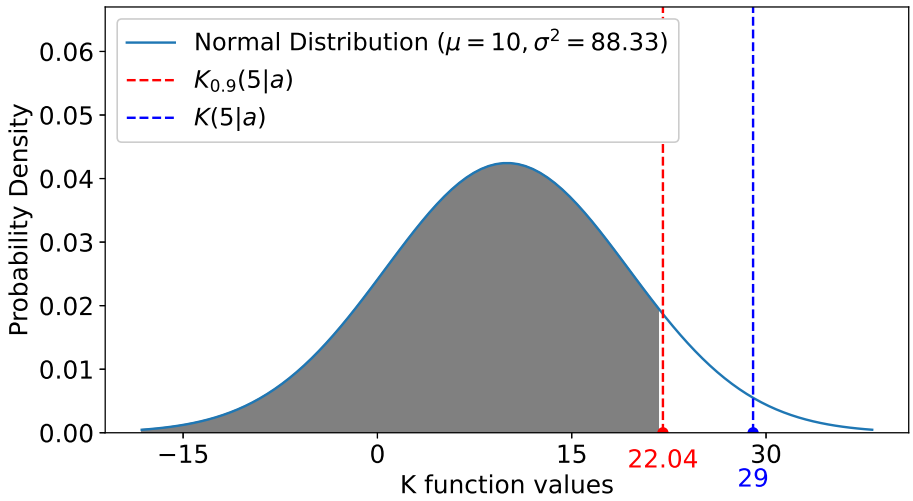


Figure 4.2: Probability Distribution Function

where α is a preset quantile threshold that serves as the statistical evidence of the identified hotspot. $K_\alpha(t|\vartheta)$ represents the α quantile of the $K(t|\vartheta)$ distribution function.

Figure 4.2 represents the CSR-based probability distribution of events surrounding Vertex a in Figure 4.1b and radius $t = 5$, given $\mu(K(5|a)) = 10$, and $\sigma^2(K(5|a)) = 88.33$. The 22.04 is the 90% quantile of the hypothetical local K-function. The actual value of the local K-function, i.e., $K(5|a)$ is 29. Hence a is considered a hotspot with confidence exceeding 90%. The quantile of a is 97.83% which represents an even higher statistical significance of the hotspot a. Higher quantile values (¿90%) make hotspot detection more stringent [125], which improves the confidence in the detected hotspots and avoids false positives [106]. Based on the above discussion, we define the following problems for hotspot detection in spatial networks.

Problem 1. Hotspot Detection with Predefined Radius (HDPR): Given a spatial network $G = (V, E)$, a minimum statistical significance threshold α, and a distance value t, find the set of event vertices V_H, such that $\vartheta \in V_H$ if ϑ satisfies $K(t|\vartheta) > K_\alpha(t|\vartheta)$.

In HDPR, hotspot detection is based on a predefined network distance, t, and a statistical significance threshold, α. Any event Vertex ϑ is identified as a hotspot if $K(t|\vartheta) > K_\alpha(t|\vartheta)$. However, in some situations, users may not have a specific idea about the best radius to choose for hotspots. Hence, we define the following problem for hotspot detection that returns hotspots alongside their radii. Instead of a predetermined radius, each hotspot is characterized by an optimal radius that maximizes the statistical significance of the detected hotspot.

Problem 2. Hotspot Detection Without Predefined Radius (HDWPR): Given a spatial network $G = (V, E)$ and a minimum statistical significance threshold α_{min}, find a set of hotspots H, each hotspot $h \in H$ is defined with a triple $(\vartheta_h, t_h, \alpha_h)$ where $\vartheta_h \in V$ is an event vertex with radius t_h and statistical significance α_h, where $K(t_h|\vartheta) > K_{\alpha_{min}}(t_h|\vartheta)$ and $\alpha_h \geq \alpha_{min}$. ϑ is considered the center of a hotspot with radius t_h and statistical significance α_h.

4.4 Hotspot Detection with Predefined Radius (HDPR)

In this section, we present our solution to HDPR problem, i.e., hotspot detection with a predefined radius. The key idea is to efficiently use the network local K-function at a specific statistical significance to detect hotspots. This requires efficient computation of N and L for every vertex in the graph. For Vertex ϑ with network distance t, $N(t|\vartheta)$

represents the number of vertices reachable from ϑ within distance t, and $L(t|\vartheta)$ represents the sum of edge weights within distance t. There are two main components of our algorithm for HDPR: (1) *Graph traversal* and (2) *Batch-Based Traversal*.

4.4.1 Graph Traversal (GT)

GT is the core technique for computing N and L values for any event vertex. GT initiates a Dijkstra [137] shortest path-based traversal that originates from an event node, say ϑ. The objective is to count events that are within a shortest path distance t and use this count to compute $N(t|\vartheta)$ and $L(t|\vartheta)$ and hence the network local K-function value for ϑ. GT starts with setting $N(t|\vartheta)$ and $L(t|\vartheta)$ to 0. Initially, the isodistance sub-network of ϑ is not discovered. GT incrementally explores edges and keeps track of all vertices (event and location) that are reachable from ϑ sorted based on their shortest path distances. Then, GT visits Vertex u that is closest to ϑ. If u is an event vertex, then $N(t|\vartheta)$ is incremented by 1. To update $L(t|\vartheta)$, GT checks the weight u can contribute to the neighboring Edge or Segment (u, v), which is $min(t - dist(\vartheta, u), w(u, v))$. If $t - dist(\vartheta, u) \geq w(u, v)$ then (u, v) is *fully traversed*. Otherwise, (u, v) is *partially traversed*. This means Edge/Segment (u, v) partially contributes to $L(t|\vartheta)$. We keep track of the partial contribution of (u, v) to $L(t|\vartheta)$, i.e., $P(u, v) = t - dist(\vartheta, u)$. GT keeps iterating until all vertices within the distance t from ϑ are visited.

Running Example: In Figure 4.1, we want to check if i is a hotspot with $t = 10$ and significance level $\alpha = 90\%$. To compute $N(t|i)$ and $L(t|i)$, we traverse the graph starting from i. j is the first vertex visited and $N(t|i)$ is incremented to 2 (from Events i, and j). $L(t|i) = 10$ because (i, j) is fully traversed and Segment (j, A) is partially traversed with

$P(j, A) = 6$. Then, Location Vertex D is visited, with $L(t|i) = 20$. This is because Segment (i, D) is fully traversed and Edge (D, G) is partially traversed with $P(D, G) = 10 - 8 = 2$. Hence $K(10, i) = \frac{N(t|i)}{\rho} = \frac{2}{\rho} = 19.33$ and that is a significance of level of 47.9%. Recall that $\rho = \frac{|V_e|-1}{L(G)} = \frac{12}{116} = 0.103$. This indicates that i does not qualify as a hotspot.

4.4.2 Batch-Based Traversal (BBT)

While GT accurately computes N and L for each event vertex, GT is inefficient as events are processed individually. We speed up GT by adding batch processing of events, i.e., Batched-Based Traversal (BBT). BBT computes N and L values of events that belong to a single Edge e in a single batch. The key idea here is that events on the same edge are close to each other and have the same neighboring events. This allows sharing the computation among events on the same edge. The objective in BBT is to compute N and L for the events along e in a single batch. This is done by traversing each edge in G and evaluating the edge's contribution to N and L of events along e. To speed up the processing of event batches within an edge, events within an edge are sorted according to their distances to one of the edge's endpoints, and the list of sorted events on e is stored as S_e. In the following, we discuss how to batch compute the N and L of events along Edge $e = (P, Q)$.

First, we evaluate the contribution of events of Edge e to their own N and L values. If $t > w(e)$, then we increment N by $|e|$, i.e., the number of events along e, and L by $w(e)$, i.e., the weight of Edge e, for all events. The reason is that all events on e are within distance t of each other. Otherwise, we apply a sliding-window-based approach that incrementally computes the N and L for each event. The length of the window is $2t$ to

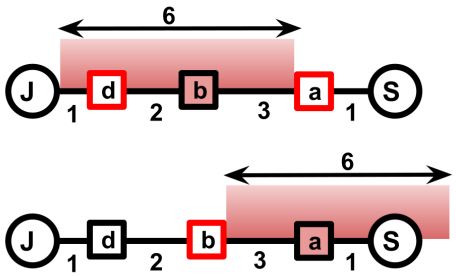


Figure 4.3: Sliding Window

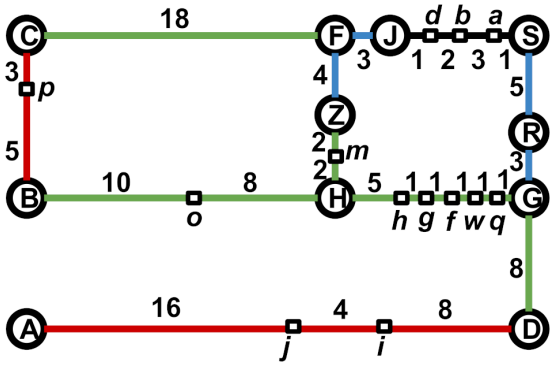


Figure 4.4: Edge Categories

account for the situation where an event is in the middle of Edge e and there are events on both sides of this event on e. Initially, the sliding window begins at the first event in S_e, say ϑ. $N(t|\vartheta)$ and $L(t|\vartheta)$ are updated according to the vertices, edges, and segments covered in the window. When the processing of ϑ is complete, the window slides towards the next event in S_e. This results in the addition of new segments to the window and the removal of other segments. The N and L of the new event can be updated according to the N, L of ϑ, and the difference in the window segments. The window keeps sliding until all events along e are processed. Figure 4.3 shows an example.

Then, we categorize any Edge, say e_1, from graph G into the following categories:

1. *All* events in e_1 are within t distance from all events in e. N and L for events on e are batch increased by $|e_1|$ and $w(e_1)$, respectively.

2. *No* events in e_1 are within t distance from events in e. In this case, e_1 does not affect the values of N and L for events on e.

3. *Some* events in e_1 are within t distance from events in e.

The processing of Categories 1 and 2 has been described above. To present our solution to Category 3, we define the notions of maximum distance and minimum distance between edges. For an Edge $\tilde{e} = (\tilde{P}, \tilde{Q})$, we define the minimum distance between e and $\tilde{e}$, as the minimum distance between any endpoint in e and any endpoint in $\tilde{e}$. Formally,

$$D_{min}(e, \tilde{e}) = min\{dist(P, \tilde{P}), dist(P, \tilde{Q}), dist(Q, \tilde{P}), dist(Q, \tilde{Q})\}$$

The maximum distance between e and $\tilde{e}$ is the upper-bound on the shortest network travel distance between any event in e and any event in $\tilde{e}$. Formally,

$$D_{max}(e, \tilde{e}) = D_{min}(e, \tilde{e}) + w(e) + w(\tilde{e})$$

In Figure 4.4, $D_{min}((H, G), (A, D))$ is the distance between D and G, which equals to 8. $D_{max}((H, G), (A, D)) = D_{min}((H, G), (A, D)) + w(H, G) + w(A, D) = 8 + 10 + 28 = 46$. For Category 3, $D_{min}(e, \tilde{e}) \leq t$ and $D_{max}(e, \tilde{e}) > t$ and some events in $\tilde{e}$ may not be reachable from events on e. This requires events in e to be examined individually.

In this case, BBT sequentially visits the events along e, i.e., $S_e[i]$ for $i = 1, ..., |S_e|$. For each event, $S_e[i]$, along Edge e, we calculate the shortest distance from $S_e[i]$ to $\tilde{P}$, denoted as $dist(S_e[i], \tilde{P})$, where $dist(S_e[i], \tilde{P}) = min(dist(S_e[i], P) + dist(P, \tilde{P}), dist(S_e[i], Q) + dist(Q, \tilde{P}))$.

Then we compute $t - dist(S_e[i], \tilde{P})$, which is the remaining weight that $S_e[i]$ can disseminate along Edge $\tilde{e}$ from $\tilde{P}$ to $\tilde{Q}$. Similarly, we compute $t - dist(S_e[i], \tilde{Q})$, which is the remaining weight that $S_e[i]$ can disseminate along Edge $\tilde{e}$ from $\tilde{Q}$ to $\tilde{P}$.

Denote $r_i(\tilde{P}) = max(t - dist(S_e[i], \tilde{P}), 0)$ and $r_i(\tilde{Q}) = max(t - dist(S_e[i], \tilde{Q}), 0)$, we define the following scenarios based on the relationship between $r_i(\tilde{P}) + r_i(\tilde{Q})$ and $w(\tilde{e})$:

(1) **Case 1:** $r_i(\tilde{P}) + r_i(\tilde{Q}) \geq w(\tilde{e})$. In this case, $S_e[i]$ can reach any place on $\tilde{e}$ within t distance. $L(t|S_e[i])$ is incremented by $w(\tilde{e})$ and $N(t|S_e[i])$ is incremented by $|\tilde{e}|$.

(2) **Case 2:** $r_i(\tilde{P}) + r_i(\tilde{Q}) < w(\tilde{e})$. In this case, $L(t|S_e[i])$ is incremented by $r_i(\tilde{P}) + r_i(\tilde{Q})$. As for $N(t|S_e[i])$, since the events along each edge are sorted, we perform a binary search with key equals to $r_i(\tilde{P})$ to retrieve the number of events on $\tilde{e}$ reachable from $\tilde{P}$ to $\tilde{Q}$, and update N. Similarly, we perform another binary search with key equals to $r_i(\tilde{Q})$ to retrieve the number of events reachable from $\tilde{Q}$ to $\tilde{P}$ and update N.

In summary, BBT creates a set $E_{e,t}$ to store edges such that $\forall \tilde{e} \in E_{e,t}$, $D_{min}(e, \tilde{e}) \leq t$. Then, BBT computes the set of location vertices reachable from P or Q within t shortest distance, denoted as $D_{P,Q}$, using the Dijkstra single source shortest path algorithm [137]. Finally, BBT iterates over location vertices in $D_{P,Q}$. For each vertex in $D_{P,Q}$, denoted as $\tilde{P}$, we add every edge connected to $\tilde{P}$, say $\tilde{e} = (\tilde{P}, \tilde{Q})$, to the set $E_{e,t}$. Then, edges $\tilde{e}$ in $E_{e,t}$ are visited incrementally and assessed for the contribution of $\tilde{e}$ towards the N and L values of events along Edge e. BBT batch update N and L for edges in Category 1 and applies a binary-search based method to update N and L for edges in Category 3. The time complexity of BBT is $O(|E_e||V_l|\log|V_l| + |E_e|^2|V_e|\log|V_e|)$. The space complexity is $O(|V_e| + |E_e|)$. The proof is given in Appendix .1.1.

Running Example: In Figure 4.4, when processing the events along Edge $e = (J, S)$, we have $D_{J,S} = \{F, J, S, Z, R, H, G\}$ when $t = 15$, and $E_{e,t}$ includes all edges except for (B, C) and (A, D). We first process e itself. Since $t = 15 > w(e) = 10$, we batch update

$N = 3$ and $L = 7$ for all events. Then, other edges satisfying Category 1, 2, and 3 are colored blue, red, and green, respectively.

After processing the edge e itself and traversing all edges of Category 1: $(F, J), (F, Z), (S, R)$, and (R, G), the N and L of events are batch updated to 3 and 22, respectively. Edges satisfying Category 2 are red-colored and pruned as they do not contribute to N and L of any event along e. Then we evaluate the edges in Category 3. Take (H, G) as an example, the minimum distance between e and (H, G) is 8, but the maximum distance is 25. Say $S_e = [d, b, a]$. We have $r_1(H) = max(t - dist(d, H), 0) = max(15 - 12, 0) = 3$, and $r_1(G) = max(t - dist(d, G), 0) = max(15 - 14, 0) = 1$. Since $r_1(H) + r_1(G) = 4 < 10$, this falls into Case 2. We invoke a binary search with a key equals to 3 from H to G to find that the number of reachable events is 0. Similarly, we invoke a binary search with a key equal to 1 from G to H and find that the number of reachable events is 1. Thus we increment $L(t|d)$ by 4 and $N(t|d)$ by 1. After batch evaluation of N and L, we have $N(15|a) = 9, N(15|b) = 7$, and $N(15|d) = 5$. Further, $L(15|a) = 42, L(15|b) = 43$, and $L(15|d) = 45$. Their respective K function values are $K(15|a) = \frac{N(15|a)}{\rho} = \frac{9}{0.103} = 87$, $K(15|b) = \frac{N(15|b)}{\rho} = \frac{7}{0.103} = 67.66$, and $K(15|d) = \frac{N(15|d)}{\rho} = \frac{5}{0.103} = 48.33$, where $\rho = \frac{|V_e|-1}{L(G)} = \frac{13-1}{116} = 0.103$. Finally, the statistical significance for a, b, d are $99.74\%, 93.64\%, 58.09\%$. Hence, a and b are hotspots with $\alpha = 90\%$.

4.4.3 Incremental Batched Traversal (IBT)

BBT requires a binary search operation for each event along Edge e to find the number of events reachable from both ends of Edge $\tilde{e}$. This operation is repeated for every

Algorithm 2: Incremental Batched Traversal (IBT)

```
Input: G : The spatial network
       t : The network traverse distance
Output: N and L for each event vertex
Ns, Ls = { }, { }
for each Edge e = (P, Q) in G do
    D_{P,Q} = location vertices reachable from P or Q within t
    E_{e,t} = Set()
    for P̃ in D_{P,Q} do
        for Q̃ in P̃'s neighbors do
            ẽ = (P̃, Q̃)
            add ẽ to E_{e,t}
        end
    end
    for ẽ in E_{e,t} do
        //edges in E_e − E_{e,t} are in Category 2 and are pruned
        if e == ẽ then
            Apply the Sliding Window approach
        else
            compute D_min(e, ẽ) and D_max(e, ẽ)
            if D_max(e, ẽ) ≥ t then
                update N, L based on Category 1
            else
                update N, L based on Category 3
            end
        end
    end
    update Ns and Ls with the events along e
end
return Ns, Ls
```

event which can be expensive. We introduce Incremental Batched Traversal (IBT) to solve this problem. IBT is based on the observation that the computations of Category 3 in BBT are repetitive and can be derived from previous computations. IBT employs incremental processing to compute N and L values for an event vertex based on the N and L values for a neighbor event.

Similar to BBT, IBT also processes events along Edge e, represented as $S_e[i]$, in sequential order, for i ranging from 1 to $|S_e|$. For each event along e, i.e., $S_e[i]$, IBT calculates values $r_i(\tilde{P})$ and $r_i(\tilde{Q})$. The handling of Case 1 in Category 3 is identical to that of BBT. However, IBT handles Case 2 incrementally.

For the first event along Edge e, represented as $S_e[i]$ with $i = 1$, IBT initiates a binary search with key equals to $r_i(\tilde{P})$ along $S_{\tilde{e}}$. Then IBT maintains a pointer, denoted as $ptr_{\tilde{P}}$. This pointer points to the furthest vertex that $S_e[i]$ can reach from $\tilde{P}$ to $\tilde{Q}$. Likewise, another binary search with a key equal to $r_i(\tilde{Q})$ is invoked and IBT maintains another pointer, denoted as $ptr_{\tilde{Q}}$, that points to the furthest vertex reachable by $S_e[i]$ from $\tilde{Q}$ to $\tilde{P}$ along $\tilde{e}$.

Beginning from $i = 2$, when computing the number of events on $\tilde{e}$ that can be reached from $S_e[i]$ (either from $\tilde{Q}$ to $\tilde{P}$ or from $\tilde{P}$ to $\tilde{Q}$), IBT starts from the previous pointers, $ptr_{\tilde{Q}}$ and $ptr_{\tilde{P}}$. It then compares $r_i(\tilde{P})$ against $r_{i-1}(\tilde{P})$ and $r_i(\tilde{Q})$ against $r_{i-1}(\tilde{Q})$. Based on these comparisons, IBT updates the positions of the two pointers, effectively tracking the current furthest event that can be reached from both ends and updating $N(t|S_e[i])$ and $L(t|S_e[i])$ accordingly. IBT continues until all the events along e have been processed. The time complexity of IBT is $O(|E_e||V_l|\log|V_l|+|V_e||E_e|+|V_e|\log|V_e|)$. The space complexity is $O(|V_e|+|E_e|)$. The proof is presented in Appendix .1.2.

Running Example: In Figure 4.4, when processing events along Edge $e = (J, S)$ with $t = 15$ regarding Edge (H, G), in the first iteration, $S_e[1] = d$, and we have $r_1(H) = 3$ and $r_1(G) = 1$, because $t - dist(d, H) = 3$ and $t - dist(d, G) = 1$. We have two pointers, ptr_H pointing to H, and ptr_G pointing to q. Then, in the second iteration, $S_e[2] = b$, and

we have $r_2(H) = 0$ and $r_2(G) = 4$, because $t - dist(b, H) = 0$ and $t - dist(b, G) = 4$. Since $r_2(H) - r_1(H) = -3$, ptr_H still points to H. On the other hand, $r_2(G) - r_1(G) = 3$ and ptr_G now points to g. Algorithm 2 gives the pseudocode for IBT.

4.5 Hotspot Detection without Predefined Radius (HDWPR)

In this section, we present the solution for HDWPR problem, i.e., hotspot detection without a predefined radius. For each hotspot vertex, there exists an optimal radius that most accurately defines the precise locality of the hotspot. In the following, we discuss how to determine the optimal radius for a hotspot.

4.5.1 Optimal Hotspot Radius Determination

Consider a hotspot Vertex ϑ, there can be $O(|V_e|)$ distinct shortest distances between ϑ to other event vertices. The possible radii for ϑ come from distinct shortest distances between ϑ to all other event vertices [108]. For example, in Figure 4.1a, there are 11 distinct distances from Event a to other event vertices. Suppose that the distances are sorted as $(r_1, ...r_{11})$ where $r_i < r_{i+1}$. For $i \in [1, 11]$, we have $r_1 = dist(a, b) = 3$ and $r_{11} = dist(a, p) = 30$. Each r_i corresponds to a statistical significance level α_i. Figure 4.5 shows the statistical significance for each radius. From the figure, we see that when $t = 5, i.e., r_2$, we achieve the first local maximum significance of 97.83%. In this case, the isodistance subnetwork of a includes a, b, and d, as shown in Figure 4.6a. When $t = 15, i.e., r_8$, we achieve the global maximum statistical significance of 97.83%. In this

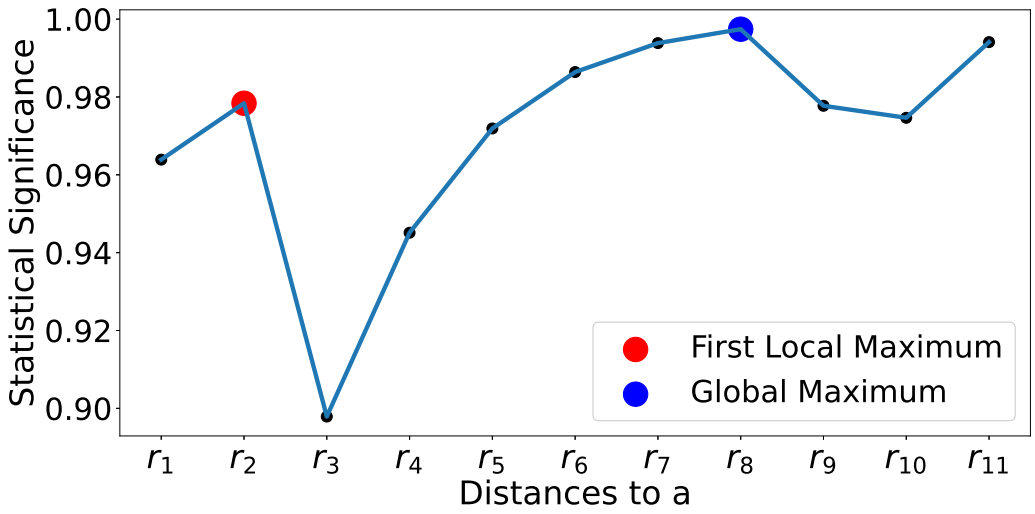
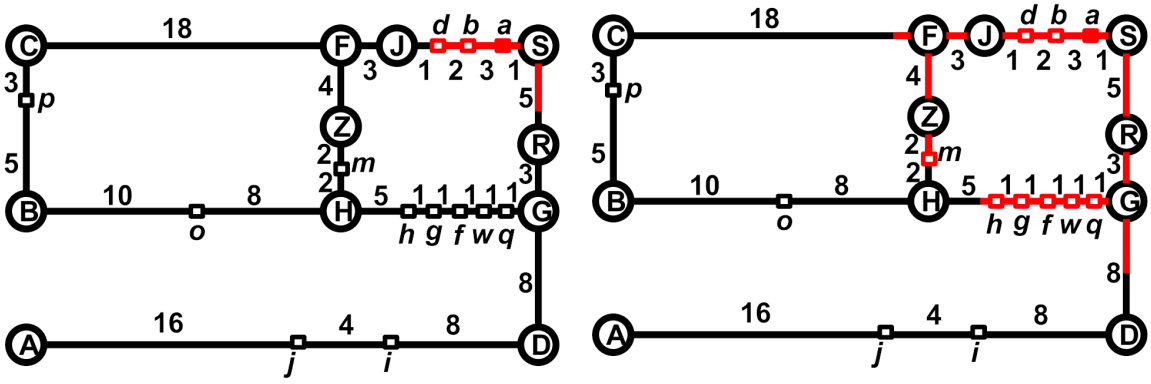


Figure 4.5: Statistical Significance under Different Radii

case, the isodistance subnetwork of a includes $a, b, d, m, h, g, f, w, q$, as shown in Figure 4.6b. The events in the isodistancce subnetwork shown in Figure 4.6a are more localized compared to the events in the isodistance subnetwork shown in Figure 4.6b.

For a given event vertex, we aim to identify the radius t that yields the ***first local maximum statistical significance***. This ensures that the identified hotspot is as localized as possible. The radius that brings the first local maximum statistical significance reflects a region where event occurrences are truly dense and not affected by other nearby clusters. This approach ensures spatial locality in hotspot detection, maximizing the relevance and accuracy of detected hotspots. This conforms to Tobler's first law of geography [138], "Everything is related to everything else, but near things are more related than distant things". Also, in information theory [139], the radius that brings this local maximum statistical significance represents the highest concentration of events, thus providing the highest information gain.



(a) Local Maximum (b) Global Maximum

Figure 4.6: Hotspot Detection Example

In the search for the optimal hotspot radius, we start with an initial radius value. Then, we increment the radius until we hit the optimal radius value. Two questions arise here, (1) what is the starting radius value, and (2) how to increment the radius. The starting radius of a hotspot needs to be neither too small nor large. If the starting radius is set too small, it could potentially lead to false positive detection of hotspots. Conversely, if we set the starting radius too large, we might overshoot and the large initial radius could encompass multiple event clusters that are not in the same spatial vicinity. Setting the starting radius to the distance between the center and the event closest to it can be problematic. The reason is that in most practical applications many events can be mapped to the same location. This is due to the low accuracy of event mapping or obfuscation of locations due to privacy constraints. In this case, the starting radius will be set to 0. This will result in a statistical significance of 1. This high statistical significance can lead to false detection of hotspots.

We set the starting radius to the weight of the edge on which the event is located. For instance, in Figure 4.6, we assign a starting radius of 7 to event a, which corresponds to

$w(J, S)$. The rationale behind this choice stems from the observation that events situated on the same edge are inherently closer and more likely to be interconnected, thus making it a suitable initial range for hotspot analysis. After obtaining the initial radius, we proceed with incrementing the radius. In each iteration, we increase the radius to the minimum value required to include the next nearest event vertex from the source event vertex. This iterative process continues until we identify a radius that results in the first local maximum statistical significance. As an illustration, in Figure 4.6, during the second iteration, we set the radius for Event a to be 10, which corresponds to $dist(a, q)$.

An exact approach to determine the optimal hotspot radius is by utilizing the Graph Traversal Algorithm (GT), which is discussed in Section 4.4.1. GT traverses the graph incrementally, visiting vertices based on their distances from the source vertex. This incremental graph traversal of GT guarantees exact and optimal results. However, GT is computationally expensive as it requires performing $O(|V_e|)$ graph traversals for each event due to the existence of up to $O(|V_e|)$ distinct distances. For instance, in Figure 4.6, the number of traversals for each event vertex can reach up to 11. Consequently, for each event vertex, this process takes $O(|V_e|(|V_e|+|V_l|)\log(|V_e|+|V_l|))$ time, since executing GT with a specific distance t requires $O((|V_e|+|V_l|)\log(|V_e|+|V_l|))$ time and there can be $O(|V_e|)$ different distances before reaching the first local maximum.

Running Example: In Figure 4.6, the initial radius for a is set to 7 because $w(J, S) = 7$. We have $N(7|a) = 3, L(7|a) = 14$, and $K(7|a) = 29$ which corresponds to a statistical significance of 91.54%. The next closest event to a is q with $dist(a, q) = 10$. Consequently, in the next iteration, the radius of a is set to 10, and $N(10|a) = 4, L(10|a) =$

22, and $K(10|a) = 38.67$, which yields a statistical significance of 89.79%. Since $89.79\% < 91.54\%$, we conclude that the optimal radius for event Vertex a is 7.

4.5.2 Approximate Hotspot Identification

To achieve a trade-off between scalability and solution quality, we propose an approximate algorithm named Approximate Hotspot Identification via Batched Edge Traversal (AH-IBT). This algorithm is an adaptation of our most efficient algorithm for hotspot detection with a predefined radius, i.e., IBT.

Similar to IBT, AH-IBT computes the optimal t for the events along the same edges in a single batch. The process involves incrementally increasing the radius values but in a coarser granularity. We gradually increase the radius to cover the nearby location vertices rather than event vertices so that the nearby edges are incrementally traversed.

While determining the optimal radii for the events along $e = (P, Q)$, we store the distinct distances between P and other location vertices in the array D_P. The array is sorted such that $D_P[i] < D_P[i+1]$ for all $1 \leq i < |D_P|$, with $D_P[1] = 0$. During the i^{th} iteration, we set the radius, t_i, for events along Edge e as $t_i = D_P[i] + w(e)$. This approach employs a coarser granularity in incrementing the radius compared to the ground truth computation, leading to enhanced computational efficiency. However, our experimental results given in Section 4.6.3 and Appendix .3.2, show that this approximation maintains a high level of accuracy. It exhibits less than a 5% error in determining the optimal radius for hotspots, effectively striking a balance between efficiency and precision.

Algorithm 3 gives the outline of AH-IBT, during the first iteration, i.e., $i = 1$, of identifying optimal radii for events along e, AH-IBT applies IBT to compute the corre-

sponding statistical significance with radius t_i for each event, C_e, where $C_e[j]$ represents the statistical significance for the j^{th} event along e. Meanwhile, we initialize a mask array M of the same size as C_e. If $C_e[j]$ surpasses the minimum statistical significance threshold in this first iteration, then $M[j]$ is assigned a true value, implying that the optimal t value for event $S_e[i]$ could be found at a larger t value. Conversely, if this threshold is not met, $M[j]$ is set to false, signifying that further processing of this event is unnecessary as it does not form a hotspot from its initial radius. The statistical significance from the initial radius should be larger than a predefined threshold, suggesting spatial locality of events is observed.

Starting from the second iteration, while recalculating the N and L values for S_e with the updated radius value, we restrict our processing to events whose mask values are set to true. Once the new statistical significance is obtained, we compare these with the previous statistical significance for events marked true in the mask array. If the new statistical significance exceeds the previous one, the mask remains true, and we update the optimal t value and the statistical significance for this event. Otherwise, the mask is set to false, and we determine the optimal t value for this event to be the last t value, corresponding to the local maximum statistical confidence. This iterative process continues until all entries in the mask array are false. The time complexity of AH-IBT is $O(|V_e||E_e|^2+|E_e||V_l|\log|V_l|+|V_e|\log|V_e|)$. The space complexity is $O(|V_e|+|E_e|)$. The proof is given in Appendix .1.3.

Running Example: In Figure 4.6, when calculating the optimal radii for the events along Edge $e = (J, S)$, the initial radius in the first iteration is set to 7. The corresponding statistical significances for vertices d, b, and a are 84.45%, 89.36%, and

91.54%, respectively. In the second iteration, the radius is set to 10 because the closest location vertex to $e[0]$ is F, and we have $t = w(J, S) + w(F, J) = 10$. With the new radius, the statistical significances for vertices d, b, and a become 81.78%, 64.37%, and 89.79%, respectively. Therefore, the optimal radius for events d, b, and a is determined to be 7.

4.6 Experimental Evaluation

In this section, we present an extensive experimental evaluation of our proposed algorithms against the state-of-the-art algorithms for hotspot detection over spatial networks.

4.6.1 Experimental Setup

We evaluate our algorithms for hotspot detection with a predefined radius (HDPR), i.e., BBT and IBT against NS*. NS* is a modified version of state-of-the-art neighborhood sharing (NS) [109] algorithm that computes the network global K-Function. While NS computes N values for each event vertex and sums it up, NS* also tracks L values per event vertex with shared processing among neighbor vertices. This allows efficient computation for the network local K-function. For detecting hotspots without predefined radius (HDWPR), we evaluate our proposed algorithm, i.e., AH-IBT against two state-of-the-art algorithms, namely, AH-NS* and NPP*. AH-NS* combines NS* with our proposed adaptive radius detection algorithm, i.e., AH from AH-IBT. NPP* is a modified version of NPP [108]. NPP is the state-of-the-art algorithm for hotspot detection in spatial networks.

NPP returns hotspots with scores assigned to all possible radii. NPP* modifies NPP to return the radius with the maximum score per detected hotspot.

We measure the runtime for algorithms that detect hotspots with a predefined radius (HDPR). For algorithms that detect hotspots without a predefined radius (HDWPR), we measure both the runtime and average error [140], i.e., t_{err}. The average error represents the mean difference between the radius computed by the hotspot detection algorithm and the optimal hotspot radius. $t_{err} = \frac{1}{n}\sum_{i=1}^{n} \frac{|t_c^i - t_g^i|}{max(t_c^i, t_g^i)}$, where t_c^i and t_g^i refer to the radius value computed by the algorithm and the optimal hotspot radius for the i^{th} hotspot, respectively. The optimal hotspot radius is defined in Section 4.5.1. We use both real and synthetic datasets in the evaluation of the proposed algorithms. The real datasets are the car collisions in the Seattle network [141], car collisions in the New York City network [142], 911 calls of the Detroit network [143], and crime locations of the Chicago network [126]. *Synth-Seattle-Random* is a synthetic dataset that is based on the road network of Seattle with randomly generated events. *Synth-Detroit-Hotspots* is a set of synthetic datasets with hotspots based on the Detroit road network topology with different numbers of location vertices and events. The parameters of the datasets are given in Table 4.1.

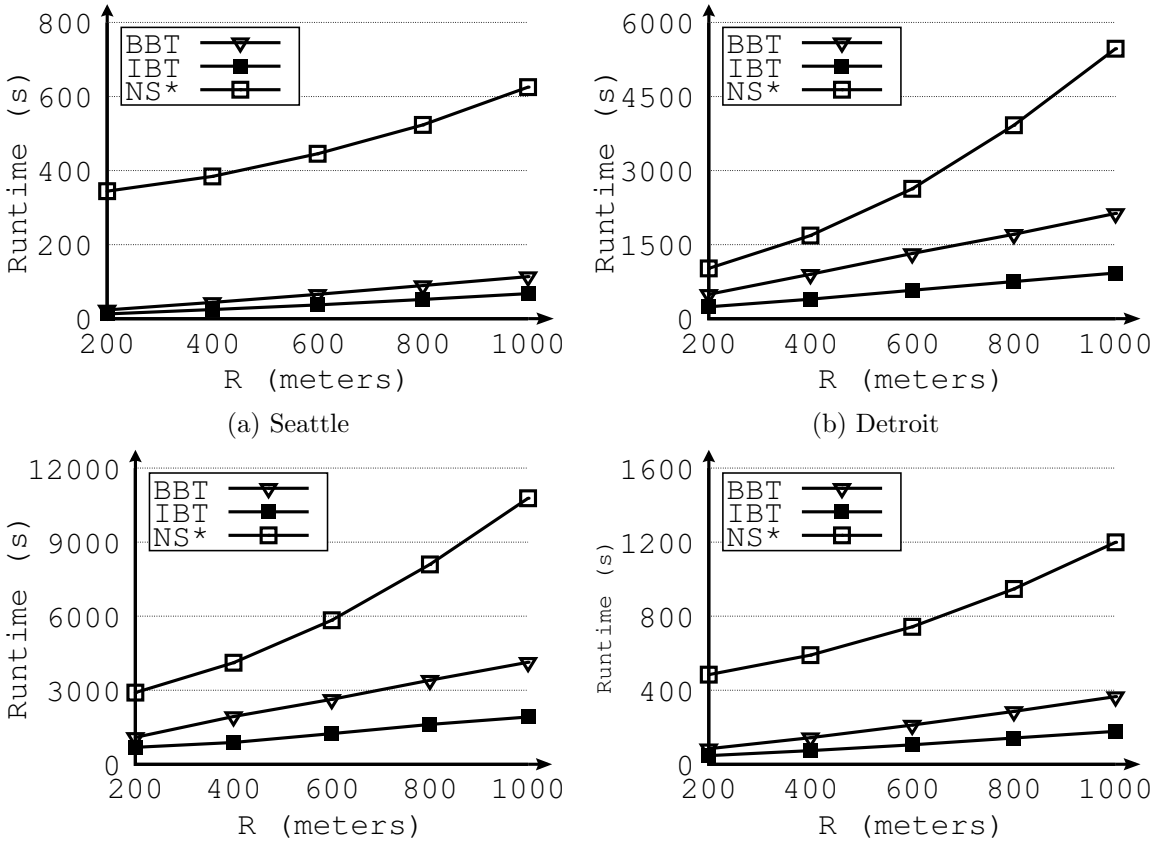
Table 4.1: Datasets

Dataset	**#Locations**	**#Events**
Seattle	11K	0.2M
Detroit	16K	5.1M
Chicago	23K	7.6M
Synth-Seattle-Random	11K	1M
Synth-Detroit-Hotspots	1 - 5K	0.5K - 500K

Table 4.2: Parameters

Parameter	**Values**
R - Radius (in meters)	200, 400, **600**, 800, 1000
SS - Statistical Significance	90%, 92%, **94%**, 96%,98%
MSS - Min Statistical Significance	90%, 92%, **94%**, 96%,98%

All algorithms are implemented using Python 3.10 and all experiments are conducted over an Intel Xeon(R) server with CPU E5-2637 v4 (3.50 GHz) and 128GB RAM running Ubuntu 16.04. Table 4.2 summarizes the parameters used in all experiments. Bold values represent the default settings for each parameter.



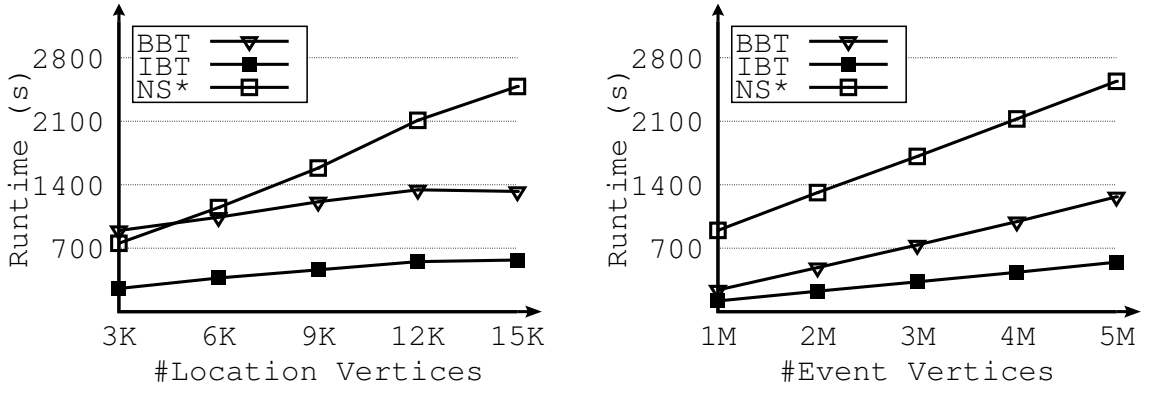
(a) Seattle

(b) Detroit

(c) Chicago

(d) Synth-Seattle-Random

Figure 4.7: Effect of Varying Radius in HDPR



(a) Varying location vertices

(b) Varying event vertices

Figure 4.8: Effect of Varying Number of Vertices in HDPR

4.6.2 HDPR Evaluation

This section evaluates the performance of BBT and IBT against the state-of-the-art algorithm NS* in computing hotspots with predefined radius (HDPR).

The effect of radius (t)

Figure 4.7 shows the runtime of hotspot detection algorithms, i.e., BBT, IBT, and NS* while changing the predefined hotspot radius under both real and synthetic datasets. In this figure, we see that IBT consistently outperforms other algorithms. IBT is up to 28.67 times faster than NS*. The efficiency of IBT is due to the following reasons: (1) efficient pruning of vertices and edges that do not contribute to hotspots and (2) batch processing of events. This batch and shared processing allow updating N and L values of multiple hotspot vertices simultaneously. Although NS* uses neighborhood sharing, NS* has a quadratic time step used to evaluate all pairs of edges in the spatial network. This step is considerably expensive and leads to a longer runtime of NS*. Also, Figure 4.7 shows that as the radius increases, the runtime for all algorithms increases. The reason is that with larger radii, hotspot detection algorithms need to explore more edges and vertices. This results in an increase in runtime. Notice that, IBT remains stable while increasing the radius. This is due to the efficiency of batch processing and neighborhood sharing used in IBT.

The effect of the number of locations

In this experiment, we investigate the effect of increasing the number of location vertices on the performance of predefined-radius hotspot detection algorithms. We generate multiple granularities of the Detroit road network using the Python OSMnx library [144] by

tuning the library's parameters. We change the number of location vertices from 3K to 15K. Figure 4.8a shows the effect of increasing the number of location vertices. IBT consistently achieves performance that is up to 4.36 times faster than NS*. Notice that, the runtime of all algorithms increases as the number of location vertices increases. The main reason is that, as the number of location vertices increases, the number of edges connecting location vertices increases. This increases the edge traversal overhead required by all algorithms.

The effect of the number of events

To study the effect of the number of events on performance, we sample events from the Detroit road network to obtain event sets with sizes ranging from 1M to 5M. Figure 4.8b shows that increasing the number of event vertices results in increasing the runtime for all algorithms. The reason is that increasing the number of event vertices results in more processing per hotspot for all algorithms. Notice that, IBT is the most efficient algorithm and consistently achieves up to 7.58 times faster compared to NS*. An additional experiment investigating how the statistical significance affects the runtime and accuracy is given in Appendix .3.1. Statistical significance has little impact on the runtime because it only affects the final validation of hotspots.

4.6.3 HDWPR Evaluation

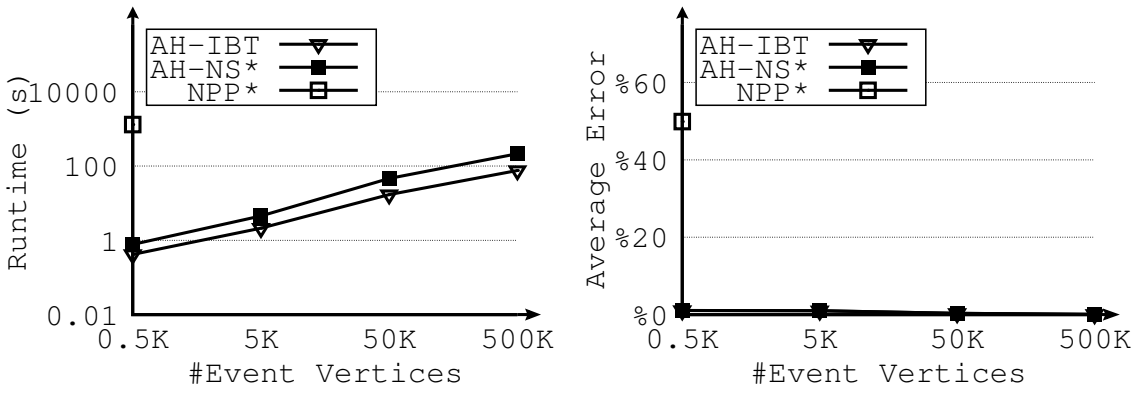
In this section, we evaluate the performance of AH-IBT against AH-NS* and NPP* for spatial hotspot detection without a predefined radius (HDWPR).

The effect of the number of events

Figure 4.9a shows the runtime of AH-IBT, AH-NS*, and NPP* evaluated with the number of events ranging from 0.5K to 500K. It is worth noting that the results for NPP* with larger datasets (events exceeding 5K) are not shown due to its extremely long experimentation time. The figure reveals that AH-IBT is more than three orders of magnitude faster than NPP* and up to 2.81 times faster than AH-NS*. The efficiency of AH-IBT stems from two main factors: (1) the inherent efficiency of IBT due to its batch processing, and (2) AH-IBT's focus on investigating hotspot radii that are likely to contribute to the final answer, unlike NPP* which performs a nearly exhaustive search on hotspot radii. Figure 4.9b presents the relative error, as defined in Section 4.6.1, for hotspot detection algorithms while varying the number of events. This figure clearly demonstrates that AH-IBT achieves up to a 50% higher accuracy compared to NPP*. The reason behind this improvement is that AH-IBT identifies the radius that yields the first local maximum statistical significance, whereas NPP* focuses more on the maximum log-likelihood ratio. Although the log-likelihood score, combined with Monte-Carlo trials, provides evidence of hotspot existence, it overlooks the spatial locality principle, which states that hotspots should be centralized and unaffected by nearby clusters.

The effect of the number of locations

Figure 4.10a illustrates the runtime of AH-IBT, AH-NS*, and NPP* using the synthetic dataset that is based on the Detroit road network with varying numbers of location nodes. Similar to the reasons discussed earlier, AH-IBT consistently outperforms the other



(a) Runtime (b) Accuracy

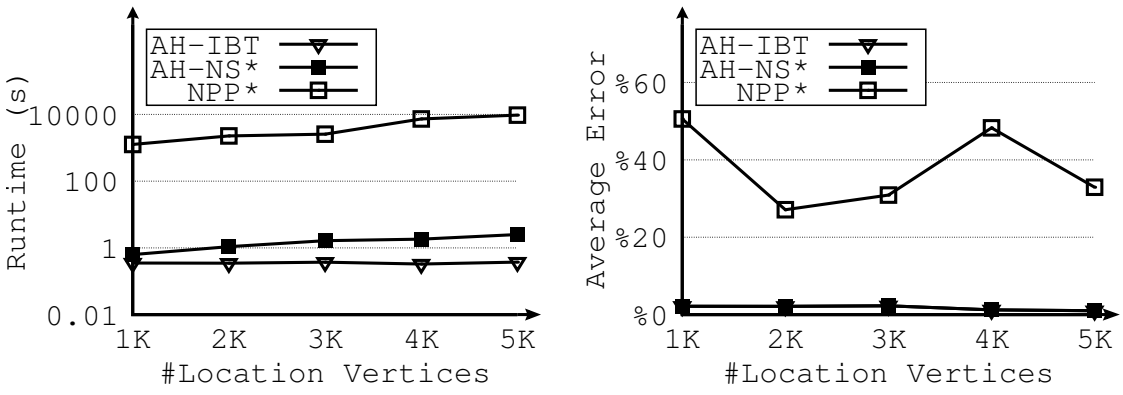
Figure 4.9: Effect of Varying Number of Events in HDWPR

algorithms, being four to five orders of magnitude faster than NPP* and 8.06 times faster than AH-NS*. As the number of location vertices increases, the runtime of all algorithms also increases due to the need to visit more edges between the expanded nodes. However, it is important to note that the performance of AH-IBT remains stable. Figure 4.10b shows the error of AH-IBT, AH-NS*, and NPP*. For similar reasons, both AH-IBT and AH-NS* achieve higher accuracy compared to NPP*, with improvements of over 50%.

Additional experiments investigating how the minimum statistical significance, the number of hotspots, and the radii of hotspots in the dataset affect the runtime and accuracy are presented in Appendix .3.2. In these experiments, AH-IBT consistently achieves the best runtime and accuracy.

4.7 Conclusion

This chapter studies scalable hotspot detection in spatial networks, ensuring statistical significance. We advocate the usage of the network local K-function for discerning



(a) Runtime (b) Accuracy

Figure 4.10: Effect of Varying Number of Locations in HDWPR

hotspots in these networks. Consequently, we present IBT, a highly efficient batch processing algorithm, specifically designed to compute hotspots over spatial networks with predefined radii. An exhaustive experimental evaluation employing both real and synthetic datasets indicates that IBT exhibits a performance that is up to 28 times faster than its contemporary counterparts. Additionally, we unveil AH-IBT, an adaptive algorithm derived from IBT, purposed for the detection of hotspots along with their radii. AH-IBT boasts a speed that surpasses the state-of-the-art algorithm by up to five orders of magnitude times and delivers an accuracy that is over 50% superior. Our future work will extend our proposed algorithms to accommodate moving objects within spatial networks.

Algorithm 3: Approximate Hotspot Identification

Input: *G : The spatial network*
MSS : The minimum statistical significance
Output: *optimal radius for each event vertex*

```
optimal_r = { }
for each Edge e = (P, Q) in G do
    mask_e = [True, True, ....] //mask array of events
    SS_e = [0, 0, ...] //statistical significance of events
    r_e = [0, 0, ...] //optimal radii of events
    t_e = w(P, Q) // starting radius
    D_P = sorted distinct distances between P to other location vertices
    while mask.any() is True do
        //process only events whose mask values are True
        SS_new = IBT(e, t, mask)
        for i = 1 to |e| do
            //process each event along e
            if First Iteration then
                if SS_new[i] > MSS then
                    SS_e[i] = SS_new[i]
                else
                    mask_e[i] = False
                end
            else
                if mask_e[i] then
                    if SS_new[i] > SS_e[i] then
                        SS_e[i] = SS_new[i]
                    else
                        r_e[i] = t
                        mask_e[i] = False
                    end
            end
        end
        incre = next smallest distance value in D
        t_e = w(P, Q) + incre
    end
    update optimal_r
end

return optimal_r
```

Chapter 5

Scalable Multi-resolution Spatial Visualization for Anthropogenic Litter Data

5.1 Introduction

Large-scale environmental problems involve managing and processing large datasets [145, 146]. A prominent example is anthropogenic litter data, which is data about waste that originates from human activities such as food waste, diapers, construction materials, used motor oil, and hypodermic needles. This waste is generated in a daily basis worldwide as a part of human daily activities. A significant part of this waste ends up in natural dumpsters, such as oceans, which causes several environmental problems [147–149], e.g., destroying marine life and increasing the impact of natural hazards like floods. This

phenomenon is becoming globally more dangerous to the extent that president of the United States (US) signed *Save Our Seas Act* in October 2018 committing the US to expand efforts to clean up nearly 8 million metric tons of litter polluting the world's oceans [150].In addition, this increasing data is affecting the quality of life in large populations urban areas, such as undeveloped neighborhoods and poor countries, through spreading diseases and health problems [151–153]. Therefore,many environmental scientists and several community and governmental organizations are interested in collecting and visualizing waste data as a preliminary step towards addressing these problems.

Environmental scientists currently use Geographic Information Systems (GIS) software to load and visualize data in the spatial space. However, with the increasing volume of the collected data worldwide, GIS software does not scale to visualize hundreds of thousands, or even millions, of data points. This limits the abilities to analyze sufficiently large datasets. Lack of such scalable visualization hinders several community efforts to clean the world, which recently encouraged interested communities to develop advanced tools to handle this [154]. Moreover, GIS software is currently limited in functionality to solve litter data issues such as cleaning noisy data and data aggregation over different attributes. This complicates integrating new litter reports and datasets, which hinders active monitoring of human litter problems and limits progress in several environmental projects.

This chapter introduces *CleanUpOurWorld*; a scalable research database that enables environmental scientists and organizations to collect, process, query, and visualize human waste data. *CleanUpOurWorld* overcomes the limitation of visualization scalability through a combination of spatial database technology and data aggregation techniques. In

addition, it employs data cleaning and integration modules that take diverse data sources, fix inconsistency problems, and integrate them into a unified logical model. It currently provides six visualization levels: continent, sub-continent, country, sub-country, city, and street levels. The street level shows individual data points while other levels show aggregate number of data points classified based on waste type as shown in our demonstration scenarios (Section 5.3). Environmental scientists and activists can interactively navigate data through zooming in/out on different spatial levels, panning over different regions, and filtering based on waste type, to visualize and analyze different portions of data with real-time response. In addition, the system allows them to add new litter data sources and download subset of existing data.

We demonstrate *CleanUpOurWorld* with an actual system implementation that integrates data from thirty different sources with total of 420K data points collected by environmental scientists and collaborator organizations. Users are able to interact with the system through different scenarios as detailed in Section 5.3.

5.2 Framework Overview

This section gives an overview about *CleanUpOurWorld* components and operations. Figure 5.1 presents *CleanUpOurWorld* architecture. The backend takes collected datasets and converts them to a common intermediate data format to standardize the input format from different data sources. Then, the converted data is fed to a pipeline of *data cleaner*, *data integrator and loader*, and *data store* that preprocesses the data and store

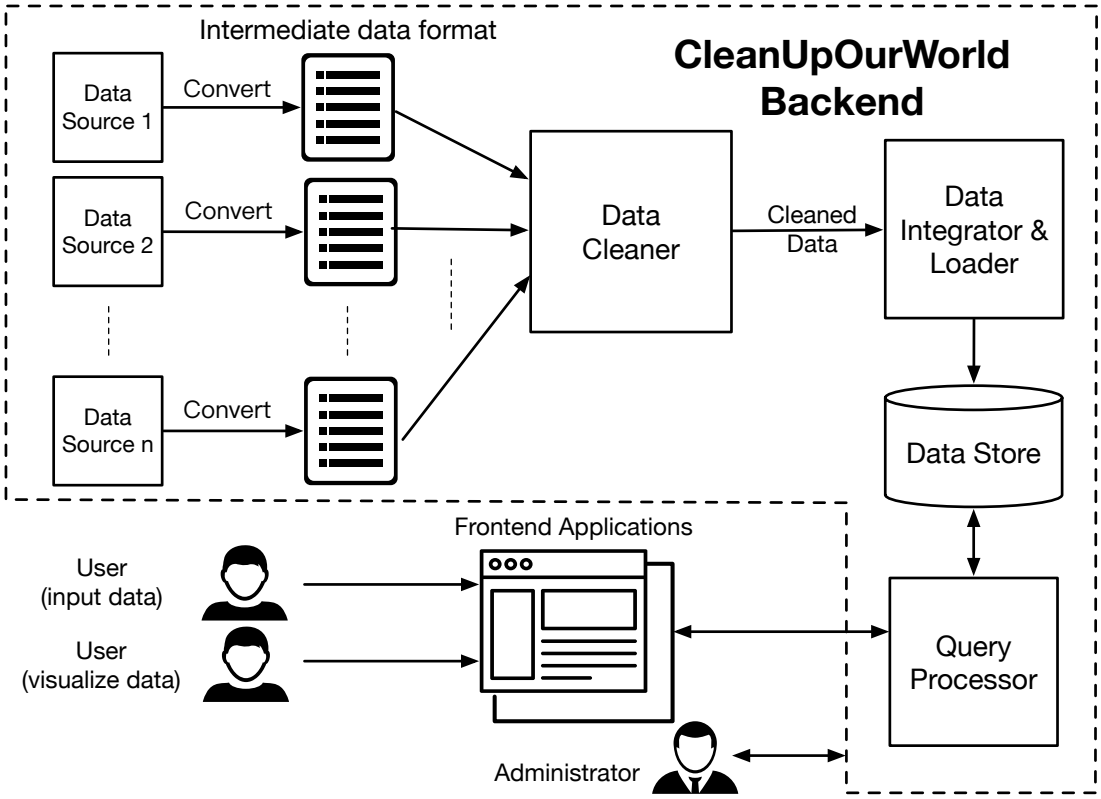


Figure 5.1: *CleanUpOurWorld* Architecture.

them in a scalable database. Finally, the *query processor* receives queries from end users through frontend applications and access the data through SQL queries. This section briefly outlines backend components and Section 5.3 outlines frontend applications.

Data collection, formatting, and conversions. In collaboration with *Let's Do it World* [155], a nonprofit organization based in Tallinn, Estonia, *Gray Lab* at the University of California, Riverside started the data collection process by contacting organizations that host the largest datasets such as *Ocean Conservancy* [156], *Litterati* [157], *Marine Debris Tracker* [158], *Alice Ferguson Foundation* [159], and *US National Oceanic and Atmospheric Administration (NOAA)* [160]. Some organizations, e.g., Ocean Conservancy and Marine Debris Tracker, provide open access to their data while others, e.g.,

Alice Ferguson Foundation and NOAA, gave us special permissions to access their data. We have collected thirty datasets so far and we are currently working on adding nine more datasets with collaborator organizations and open data sources. In addition, we are running a crowd-sourcing data collection project in Southern California. Datasets are collected in four formats: ESRI Shape files, KML files, XLSX files, and CSV files. Then, all formats are converted into a common CSV format with UTF8 encoding.

Data cleaning. *CleanUpOurWorld* employs a data cleaning process that addresses litter datasets collected from diverse sources. The process includes three operations. First, regulating columns names to avoid empty names, duplicate names, mismatched names, and names that do not comply with SQL standards. The problem of mismatched names arise when a common essential attribute, such as location, appears with multiple names. We detect this for location and date attributes through comparing both data types and names and fill up with a common attribute name that is used in all datasets. Other than mismatched names, empty, duplicate, and non-compliant names are fixed with arbitrary names, through regular expressions, prompting the user asynchronously to alter them with new names if needed. Second, fixing data irregularities where some data values are not compliant with the data type and in other cases values of the same attribute has different formats. Regular expressions are used to discover non-compliant values and unify the format of the same data type. Non-compliant values are filtered out for further manual processing so that the column data type can be assigned the appropriate data type. Third, some attribute irregularities go beyond simple format mismatches to completely different format. For example, the location attribute is an essential attribute in all our datasets

and its common format is a pair of latitude/longitude coordinates. However, some datasets have this attribute as a rough textual description, such as street name, city name, etc. To address this, we employ a Python library GeoPy, with *Nominatim* geocoder, that converts this text into precise coordinates.

Data integration, loading, storage, and visualization. After each dataset is cleaned, it is forwarded to a data integrator and loader module. This module goes over three steps. The first step loads the dataset in a separate SQL table, which contains all input data attributes, in a PostGIS database that represents the main data store in Figure 5.1. The loading process is performed through a dynamic SQL function, using *plpgsql* language. Then, it creates a SQL table, populates the data from the file to this table, and adds an auto increment primary key field that guarantees a unique identification for each record in the system. The second step loads only essential attributes of each record from the newly loaded dataset in a centralized table, called *maintable*, that integrates data from all existing datasets in a snowflake-like fashion. This table is created to scale up query processing as we will elaborate later. The *maintable* have three essential attributes per record: a location as latitude/longitude pair, a date, and a collecting organization, in addition to the record id and the dataset name to link the record back to its original dataset table with full attribute set. The *maintable* has one record corresponds to each record in each dataset. The third step aggregates data from the *maintable* in the *aggregatetable*. When environmental scientists visualize data in large regions, e.g., city, country, continent, or ocean, the large number of points in the region will limit the existing visualization frontends, such as GoogleMaps and GIS software, to show all points while still being interactive to users. To overcome

this problem, *CleanUpOurWorld* visualizes individual data points only at the street level, while aggregate data is visualized on higher levels. To speed up such aggregate visualization, *aggregatetable* maintains aggregate counts from *maintable* at five different spatial levels that corresponds to the frontend visualization levels. At each level, the whole world is divided into a set of spatial tiles. Then, the *aggregatetable* maintains the number of data points in each spatial tile classified by the litter type. When a visualization query comes, this table is queried to retrieve data right away and visualize them to users.

Query processing. The query processor receives three types of queries from frontend applications and accesses the database to return their results. First, a spatial query that finds all data points in a certain spatial range. This query is answered from *maintable* with a single spatial range query. Second, a spatial query that finds aggregate counts in a certain spatial range. This query is answered from *aggregatetable* with a single spatial range query as well. Third, a query that finds all attributes of a single data record. This query is answered from the corresponding dataset table with a single query given the record id. Using *maintable* and *aggregatetable* enables each incoming query to be translated into a single SQL query, which allow scalable management and visualization of litter data.

5.3 Example Use Cases

CleanUpOurWorld functionalities and internals are demonstrated using the first batch of real datasets collected from thirty different sources with total of 420K data points. New data batches with hundreds of thousands of points are being collected as described

earlier. Our users would be able to interact with *CleanUpOurWorld* through one or more of the following example use cases.

5.3.1 Use case 1: Hypothesising and Locating Litter Through Interactive Visualization

The main objective of *CleanUpOurWorld* is to enable environmental scientists and community organizations to address human litter problems through analysis, proposing hypotheses, or locating data to clean up certain places. Thus, users are able to interactively visualize litter data based on multiple dimensions: different spatial levels, arbitrary time periods, categories of different litter types, and collecting organizations. Figure 5.2 depicts the main visualization screen of *CleanUpOurWorld*. Figure 5.2a depicts the highest spatial level of visualization. At this level, the whole world is divided into ten large regions based on continents and their adjacent oceans, eight of them are shown in the figure. The figure shows the spatial boundaries of each region. In addition, each region has a single pie chart that shows percentages of each litter type, out of ten types depicted in the legend. On hovering over the pie chart, users are able to know the actual number of points of each litter type. Zooming on the map to deeper levels shows a pie chart for each sub-area, either in oceans or in continents. This enables scientists and activist to narrow down areas of interest based on certain litter types, e.g., determining that the northern part of the Pacific ocean witnesses the highest plastic pollution.

The environmental scientist can zoom in and out on the map view to show finer granular data on six different levels: continent, sub-continent, country, sub-country, city, and street level. By zooming in/out on the map view, the data is automatically divided or

merged to show data that corresponds to the current level. The sub-continent level shows data from multiple countries in one spatial tile and divides the whole world into 10x10 tiles, and so on up to the street level. Figure 5.2b depicts the street level visualization in Riverside, California. At this level, only individual data points are shown without any aggregation. The viewed data subset is reflecting the applied filters of litter type, time period, and collecting organizations.

5.3.2 Use case 2: Adding a New Litter Data Source

The on-going data collection process and the natural continuity of human litter necessitates adding new datasets to the database. A simple way is adding new data manually through the administrative tools, e.g., PostGIS admin tools. However, this will limit both number of collaborations and amount of data that contribute to our repository. To make it a comprehensive research database that serves as many environmental scientists worldwide as possible, we are enabling external data entry so that scientists and organizations can contribute their data to our repository easily. Figure 5.3 depicts a data entry screen that enables uploading external data to *CleanUpOurWorld*. The screen allows the user to upload a file of a certain format. Currently supported formats are ESRI Shape files, KML files, XLSX files, and CSV files. By default, the first data row is considered attributes names. After the file is successfully parsed and loaded, the user can rename attribute names and determine their data types. In addition, she can edit any data value in any row so she can possibly add any missing values or correct any incorrect values. Once all corrections are made, the user can submits the uploaded data to our backend database.

The newly uploaded data is not directly integrated with our existing data. Instead, the new data is loaded in a new separate database table and the system administrators are automatically notified with a new dataset addition request. Then, the request is reviewed for data adequacy in terms of completeness of necessary attributes, matching attribute names, and lack of any data problems such as SQL injection data, severe noise data, etc. In several cases, it is needed to follow up with the data owners to fix data problems before allowing the new data to be integrated with the existing data. Once the uploaded data is put into a good shape to be integrated, the cleaning, integration, and loading process of *CleanUpOurWorld* is executed so the new data is processed as part of our database. Although this process is lengthy and might include several cycles of interactions between database administrators and data owners, it still allows much faster process than individual collaborations as uploading and refining the data is much faster.

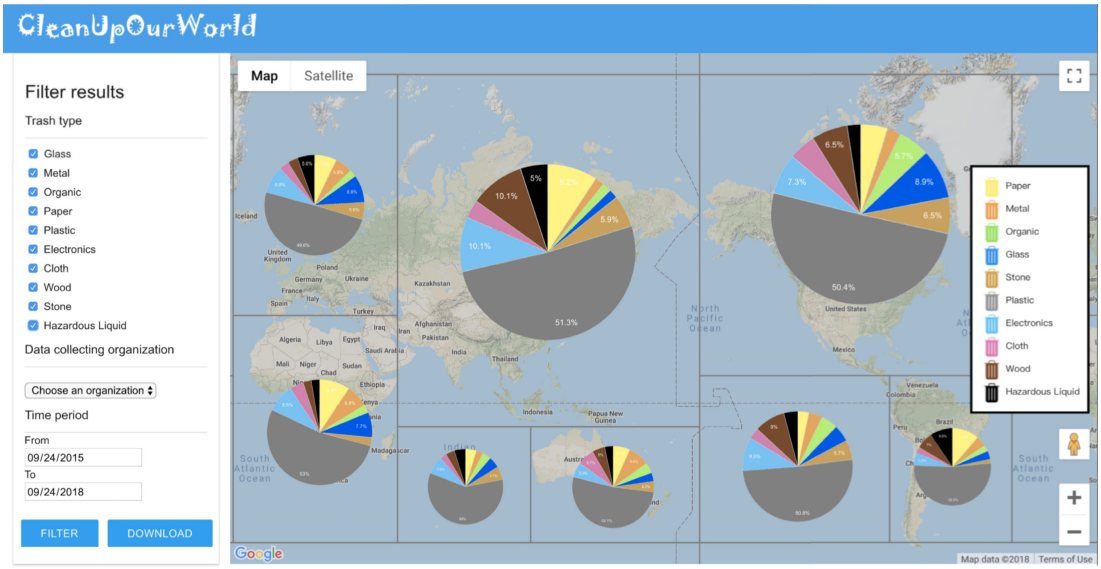
5.3.3 Use case 3: Downloading Litter Data

CleanUpOurWorld will have its prominent value with enabling environmental scientists to use it as data repository where they can *add* and *get* data in addition to being a scalable data manager and visualizer. This open culture encourages the environmental scientists and community organizations to invest time, efforts, and contribute data to this repository, which significantly enrich our collaboration network. For this, we will enable users to download subsets of data based on terms and conditions of each dataset. Figure 5.2 shows a download button that enables demo attedees to download the subset of data that is currently visualized on the map view with all the applied filters. The downloaded data will include any aggregate data that is shown on the map view, in addition to any individual

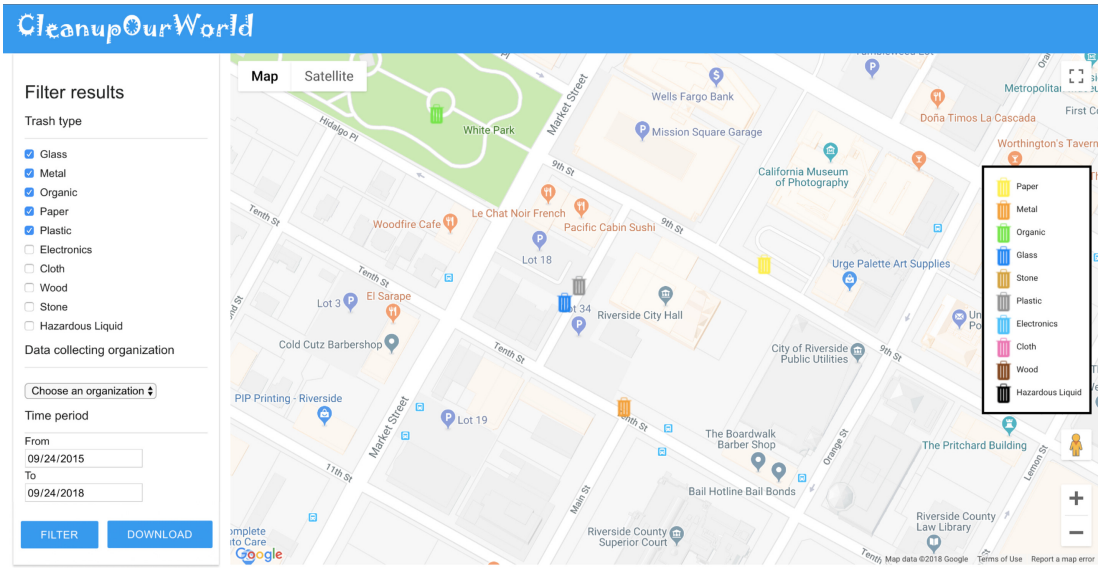
data points, contributing to these aggregates, that are permitted by the data owners to be downloaded.

5.3.4 Use case 4: Administrating Existing Litter Data

Despite the automated tasks provided by *CleanUpOurWorld* in maintaining litter data, database administrators may still need to manually fix data issues, e.g. clean unknown formatted attributes or tweak any dataset-specific problem. Thus, users are able to interact directly with the backend database via the administration console through either command line or GUI interfaces. This includes executing SQL queries of different types. Examples are selection queries to view existing data records, update statements to modify certain data values, alter table statements to modify data scheme properties (changing attributes names, data types, adding attributes, etc). In addition, users are able to interact with dynamic SQL functions that are used in several operations such as data loading, part of the cleaning process, etc.

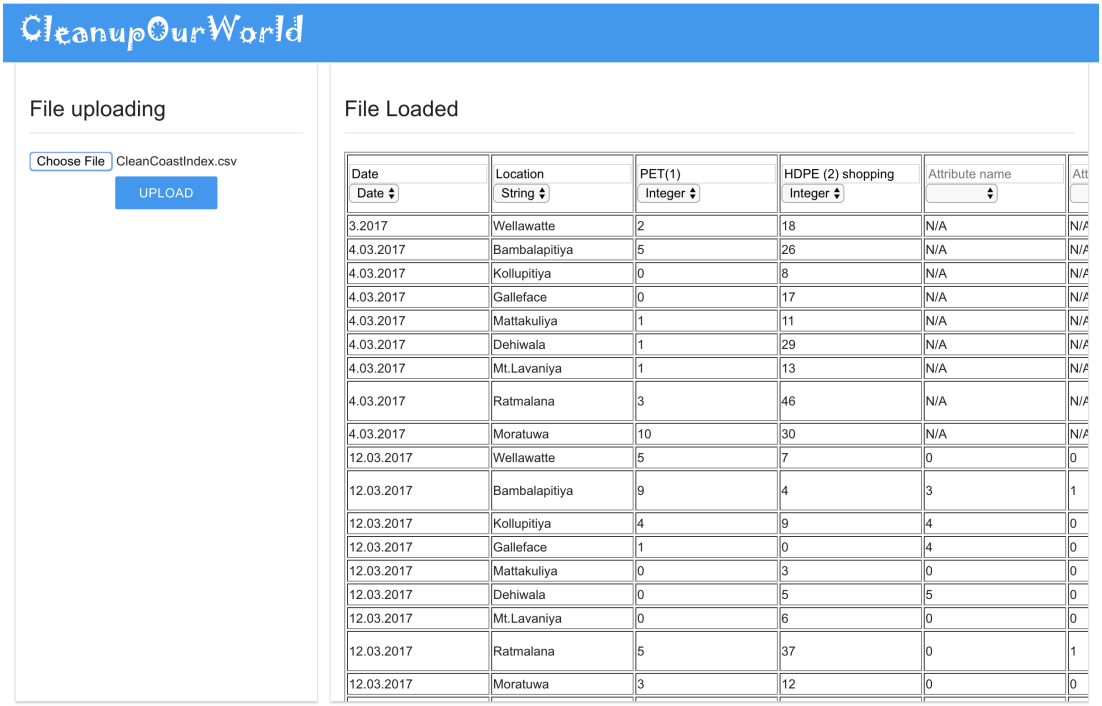


(a) Litter Data on Continent Level



(b) Litter Data on Street Level

Figure 5.2: Visualizing Litter Data on Multiple Spatial Levels.



Date	Location	PET(1)	HDPE (2) shopping	Attribute name	Att
Date	String	Integer	Integer		
3.2017	Wellawatte	2	18	N/A	N/A
4.03.2017	Bambalapitiya	5	26	N/A	N/A
4.03.2017	Kollupitiya	0	8	N/A	N/A
4.03.2017	Galleface	0	17	N/A	N/A
4.03.2017	Mattakuliya	1	11	N/A	N/A
4.03.2017	Dehiwala	1	29	N/A	N/A
4.03.2017	Mt.Lavaniya	1	13	N/A	N/A
4.03.2017	Ratmalana	3	46	N/A	N/A
4.03.2017	Moratuwa	10	30	N/A	N/A
12.03.2017	Wellawatte	5	7	0	0
12.03.2017	Bambalapitiya	9	4	3	1
12.03.2017	Kollupitiya	4	9	4	0
12.03.2017	Galleface	1	0	4	0
12.03.2017	Mattakuliya	0	3	0	0
12.03.2017	Dehiwala	0	5	5	0
12.03.2017	Mt.Lavaniya	0	6	0	0
12.03.2017	Ratmalana	5	37	0	1
12.03.2017	Moratuwa	3	12	0	0

Figure 5.3: Adding a New Litter Data Source.

Chapter 6

Unified Data Model for High-definition (HD) Map Data

6.1 Introduction

Existing mapping platforms, e.g., Google Maps, are designed, built, and deployed for human-to-machine interaction, where the major implicit assumption is that a human user is consuming the mapping data from a computing device, e.g., a smartphone. Over the past few years, the rise of self-driving cars technology has triggered an urgent need for a new type of mapping data that is designed, built, and deployed for machine-to-machine interactions as the human driver is being replaced with an automated machine driver. This is the main motivation behind the research wave of "high-definition" maps or "lane-level" maps, that includes several research efforts [161–167] to enrich existing maps with much more detailed information that are needed to be consumed by machine users.

High-definition map (HD map) provides accurate and detailed lane-level 3D information and it plays an essential role in several important applications for self-driving cars. The detailed 3D localization objects included by HD maps enable the self-driving cars to have a more precise and comprehensive knowledge of its environment and help the vehicles to better determine their position and orientation, which enables the autonomous vehicles to perform controlled maneuvers beyond the sensing range [168]. Potential risks and obstacles ahead that are out of the range for the sensors can also be predicted and avoid with the rich information [169]. HD maps are also able to store prior information that are of great help when the self-driving cars need to perform computations for applications such as traffic light recognition and speed sign recognition [170]. In addition to the application and techniques required by self-driving cars, HD maps are also deployed to help with the development of the advanced driver assistant systems (ADAS) and push the existing navigation applications to provide more detailed lane-level instructions. The development of these techniques will improve the driving experience of human users as well.

The huge potential of HD map has attracted plenty of efforts on map data collection, automated map modeling and algorithm design for related applications. However, there are arguments that the existing HD map data formats are not research-friendly. The existing commercial HD map models are confidential, and some require a commercial membership to be accessed. The commercial models, taking the Navigation Data Standard (NDS) as an example, are complained to be not suitable for researches due to their complexity and the features supported [171]. Researchers with such complains tend to define their own data model. The different data models used in different research works may down-

grade the reusability of the data and hinder the experiments that need to be performed on different data sources. Thus, we propose to design a standard research-friendly data model for HD map data.

Our aim of designing the data model for the roads is to extend the node-edge structure. There are three goals that we want to achieve with this model:

The first goal is to design a research friendly standard for HD map data. Different map providers have different definition for HD map data [168]. In addition, the leading data format for HD maps, the NDS format, is identified to be not "research friendly" enough due to its complicated data structure and the features supported [171]. Hence, there is a need for a standard research friendly HD map format.

The second goal is to maintain the widely used Intersection-Edge model. In this way, the existing information and applications associated with the traditional maps can be safely imported to our model without worrying about compatibility.

With the two goals in mind, we define our data model. The model consists of two major parts: road network and landmarks. We enrich the information for road network by extending the Intersection-Edge model using Lane Bundles while maintaining the compatibility for existing maps. The Landmarks are identified to record the 3D objects in the driving environment to facilitate the vital applications for autonomous driving cars such as localization.

The remainder of this chapter is structured as follows: Section 6.2 reviews the commercial and research-oriented HD map models we have knowledge about. Our data

model design is described in Section 6.3. We provide a discussion in Section 6.4 and we conclude with perspective and future work in Section 6.5.

6.2 Related Work

Geographic Information System (GIS) is playing an important role in the development of smart cities and there have been discussions about building 3D-GIS maps for smart management and decision making [172–174]. In this chapter, we focus on the HD map for autonomous vehicle applications. HD map for commercial purpose was first launched in 2015 by Tomtom [175]. Since then, several map providers have defined their own HD map standard. Navigation Data Standard (NDS) is the leading data format in the industry [171]. The NDS format classifies all data into different building blocks, such as routing building block, lane building block and 3D objects building block [176]. Building blocks are used to address different functional aspects of NDS. The TomTom HD Map with RoadDNA model is also claimed to be a highly accurate representation of the road. It provides numerous features, including lane models, traffic signs, road furniture, and lane geometry, with high accuracy [175].

Although being supported by major industry, there are some potential drawbacks for existing commercial data models for HD maps. One main concern is being confidential and not open for research purpose. It is hard to obtain the data as well as the data format specification. From a research perspective, there are scattered data that is valuable for constructing partial HD maps, such as the OpenStreetMap data and other crowdsourced data sources. However, the lack of an open standard data format adds difficulty for using

such data and also downgrades the reusability of them. Secondly, the features supported by the data models tend to be demand-driven. The features included are for achieving certain functions or benefits. This may be helpful for supporting different applications, but it also makes the features to be discrete and set obstacles for introducing new features for researchers, especially those similar to or closely related to the existing features in the data model. Last but not least, commercial data formats are usually divided into discrete building blocks or layers and this design choice should be closely related to the objectives of modularity, compatibility, and interoperability. However, applications that make use of the HD map may require specific types of data. As a result, the data models tend to have detailed divisions that are on the same level and become too complicated for research purpose.

For all the above reasons, we observe that researchers tend to start from a relatively simple model and focus on certain features when studying HD maps. Jiao pointed out that the four elements an HD map usually contains are the reflective map of the road surface, the 3D shape of the surface of the driving environment, lane/road model and the static elements in the driving environment [162]. Some researchers from HERE, a major map provider worldwide, construct a road model containing only lane boundary, occupancy grid and the road sign [171]. Takeuchi et. al. make use of Point Cloud Map and Vector maps for blind traffic prediction [169]. The researches on HD map creation also focus on determining the details of the lanes [161–166] and modeling the driving environment [165, 167]. Hence, we may draw a conclusion that the basic components of HD map data models for research

should be the road, which is extended to the level of lanes and the peripheral driving environment.

6.3 Data model

In this section, we give descriptions for our HD map data model. Our model builds on the popular node-edge model that has been used for long in academic research environments. The node-edge model represents the road network as a set of nodes and edges. Edges represent road segments and nodes represent intersections between roads. Special semantics are added to handle special cases, such as dead ends and different types of intersections, e.g., 2-way. 3-way, or 4-way intersections. This model is intended to be simple, of low storage, and isolates road network data from all other data objects in the outdoor space due to lack of applications that comprehensively use all these data items together.

The standing point of our model is to extend the node-edge structure to model the new rich information added to HD maps. The new information includes the lane system and various types of landmarks. This modeling effort tries to provide a research-friendly model for HD map data, compared to the existing various models [168] and the NDS standard that is argued not to be research-friendly enough due to its complicated data structure, the features supported, and openness [171] as discussed in Section 6.2. In addition to regular road network information, HD maps also include landmarks data that is essential for several applications, such as localization as well as obstacle avoidance for self-driving cars.

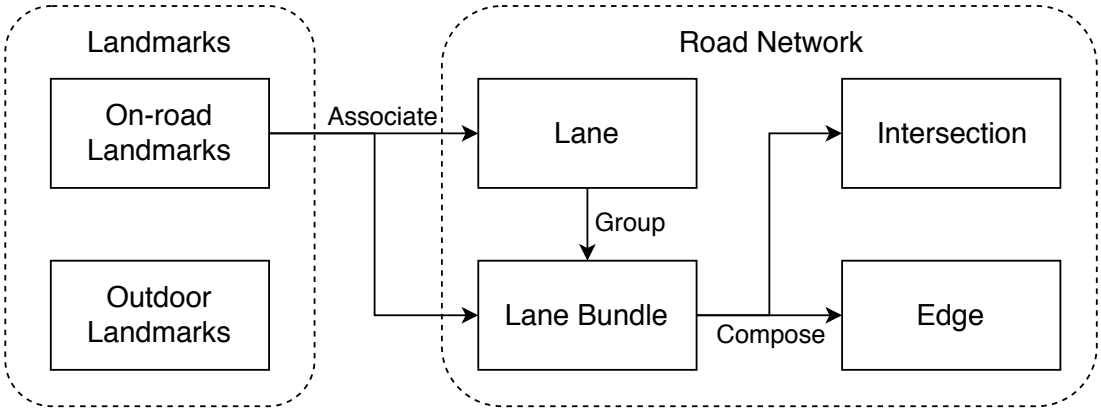


Figure 6.1: Overview of the data model

Figure 6.1 depicts an overview of the data model. The model schema is divided into two parts, the road network, and the landmarks. The road network still consists of two main entities, Intersection and Edge, that correspond to node and edge in the traditional model and represent intersections and road segments, respectively. In addition, both entities are composed of LaneBundle entities that represent the lane-level information of the road network. Each LaneBundle consists of a group of Lane entities. Both Lane and LaneBundle entities can be associated with on-road landmarks that are represented by On-road Landmarks. Examples of on-road landmarks are road signs and traffic lights. In addition, landmarks could also include information about off-road landmarks, represented by Outdoor Landmarks in the figure. Details of both road network and landmarks are described in the rest of this section.

6.3.1 Road network

The roads in traditional maps can be viewed as intersecting polylines. The two central elements of the road network data model are Edge and Intersection, which corre-

spond to the edge and node in the popular node-edge model. Our model extends these two elements with lanes and lane bundles. We will first describe the definition of the lane and lane bundle and then show how we extend the Edge and Intersection to model the HD map.

A lane, corresponding to the regular driving lane, is a sectional slice of the road segment. It is modeled as a 3D surface with a boundary that records the information including the basic shape of the driving lane and the elevation. Each lane starts and ends with a cross-sectional boundary, which is a three-dimensional line in the space, usually perpendicular to the lane driving direction, starts and ends with a node that is a three-dimensional point in the space and represents the farthest corner point in the lane. A lane reference is associated with each lane to help to show the orientation of the lane and locate the point attributes and range attributes. Example of point attributes is the vehicle checkpoints where polices may stop the vehicles to conduct the roadside detention with a legitimate reason. Example of range attributes is the lane maximum speed that might have different values over different distance ranges in the lane, e.g., the lane has a maximum speed of 30 MPH before a certain point and 40 MPH after that. Parallel lanes in the same road segment are grouped into lane bundles. A lane bundle reference is used to record the attributes of the lane bundle and the attributes would apply to all the lanes within the current lane bundle. The lane bundle starts and ends with a cross-sectional boundary, represented as a polyline, that is the union of all cross-sectional boundary lines of all lanes within the lane bundle.

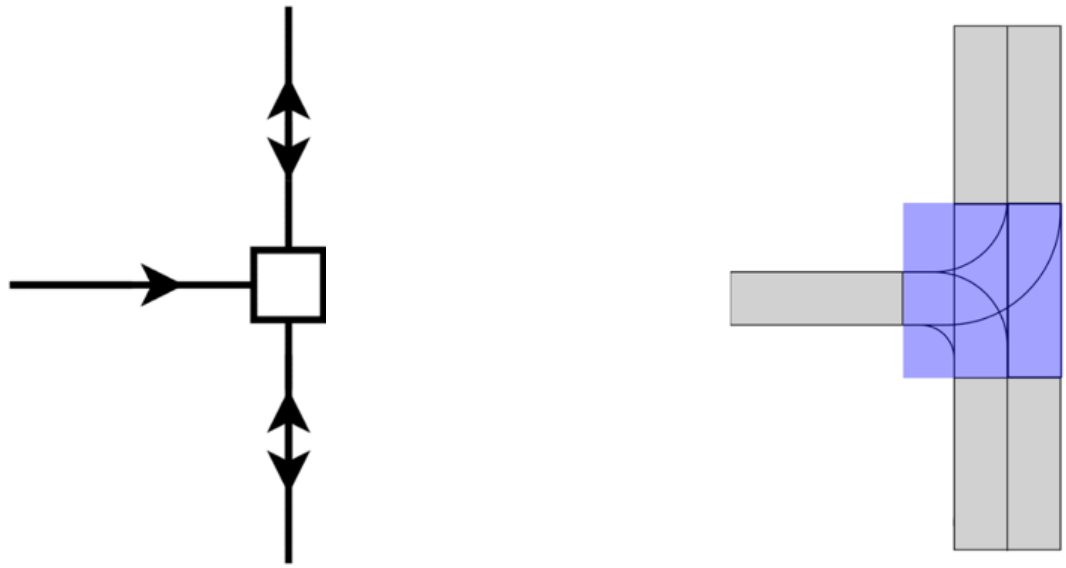
With the definition of lane and lane bundle, we then extend the node-edge model in the traditional map models. In the traditional maps, a road segment is represented by

an edge that is a simple polyline. We extend the Edge into a sequence of one or more connected lane bundles. The connection between two lane bundles is represented with sharing the same cross-sectional boundary polyline. Thus, two lane bundles B1 and B2 are connected if and only if the starting boundary of B2 is the ending boundary of B1 or vice versa. Furthermore, the nodes in the traditional map represent the intersections among road segments, i.e. Edges. In our model, an intersection, represented by the entity Intersection, is a group of overlapping lane bundles in different directions based on the type of intersection, e.g., 4-way, 3-way, or 2-way intersections. Each lane bundle in the intersection connects two lane bundles that belong to two different road segments, either in the same direction or in angular directions, e.g., perpendicular directions. Figure 6.2 gives an example of the Intersection-Edge representation and the lane bundle representation of the same T intersection of a one-way street and a two-way street. Lane bundles boundaries are represented by the solid lines in the right figure. The area bounded by the blue rectangle corresponds to the Intersection and the gray lane bundles correspond to the Edges.

The Intersection-Edge model is extended using the lane bundles so that the HD map can carry more detailed information for emerging applications while maintaining the practicality. The simplicity and the potential for feature extensions enable the data model to be more research-friendly.

6.3.2 Landmarks

Landmarks are intended to include the objects in the driving environment. We classify the landmarks into two subclasses: on-road landmarks and outdoor landmarks.



(a) Edge-Intersection (b) Lane Bundle

Figure 6.2: T intersection of a one-way street and a two-way street

On-road landmarks represent landmarks that are associated with either the road segments or intersections and are part of the driving roads, such as lane marks, traffic lights, road signs, and guardrails. The on-road landmarks carry certain attributes or will lead to the changes of attributes of lanes or lane bundles on certain directions. For example, lane marks will determine whether a vehicle can pass certain lane boundaries to turn or overtake another vehicle and speed sign marks change of the speed limit attribute for lanes or lane bundles at a certain point.

Outdoor landmarks represent other real objects that exist in the outdoor space but not part of the road network, such as buildings, trees, parking lots, lights, and parks. These objects are diverse and of different nature and could have different usages for HD map consumer. For example, trees and lights could be used for self-localization algorithms while parking lots and parks could be used as routing destinations.

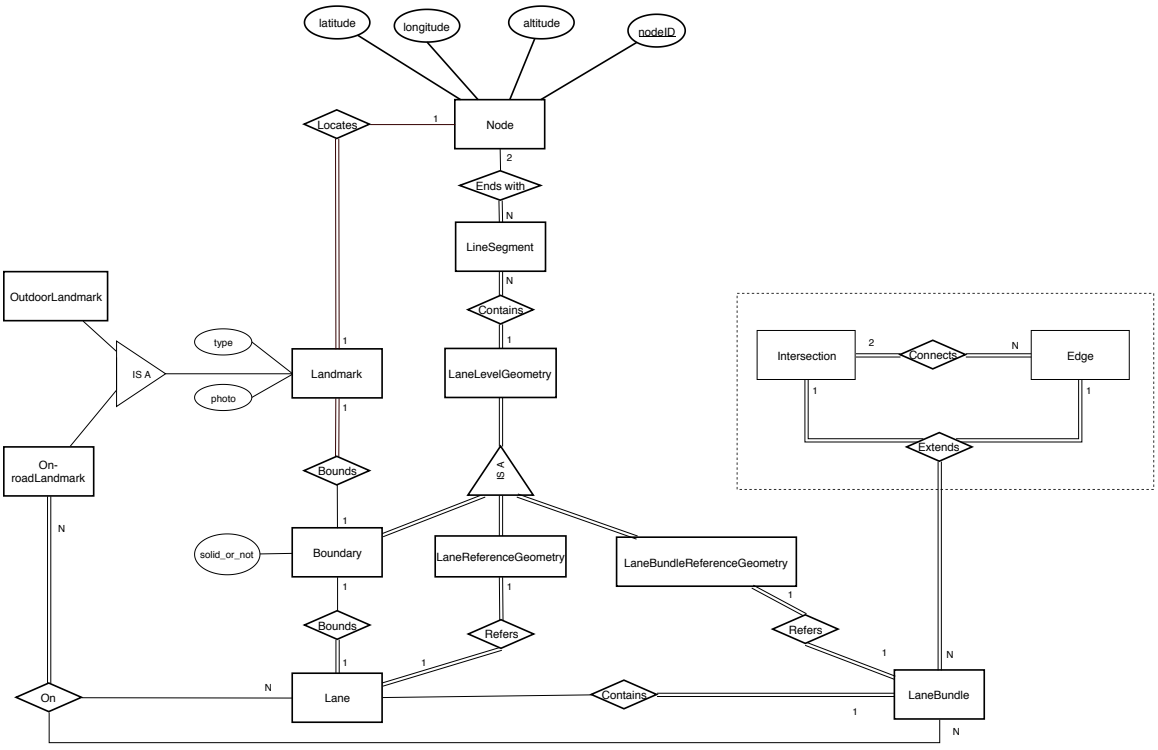


Figure 6.3: Data model

Both on-road and outdoor landmarks are represented by Landmark entity that has generic attributes such as type and photo. The generic representation of a landmark comes with the reality of limited data available on this type of entities for now, so generality allows more flexible accommodation for new information later.

6.3.3 Detailed Schema Description

The previous subsections give a high-level overview of different components of our proposed model. This subsection gives a detailed data scheme along with a more detailed description of different entities and relationships among them. Figure 6.3 shows the data schema. We start from the Intersection and Edge model and extend them with the LaneBundle to accommodate lane-level information. The LaneBundle is defined to be a

group of parallel Lanes. To register and locate the attributes on Lanes and LaneBundles, a reference geometry is associated with each of those entities. The reference geometry represents the underlying physical spatial coordinates, represented with points, lines, and polylines of latitude and longitude coordinates. Points are modeled with entity Node, lines and polylines are modeled with the entity LineSegment, and more complex geometries are represented with the entity LaneLevelGeometry. Each LaneReferenceGeometry or LaneBundleReferenceGeometry has an orientation that represents the orientation of the Lane or the LaneBundle. An attribute is registered using the relative distance from the start of the reference geometry. An attribute could be a point attribute, such as a U-turn point or a vehicle checkpoint, or a range attribute, such as a range on the road for the speed limit. Each Lane is bounded by a Boundary. The Boundary is associated with the attributes that depict its characteristics, such as whether it is a boundary that connects two Lanes from different LaneBundles or two parallel lanes in the same LaneBundle, whether it allows vehicles to pass through it to perform maneuvers. The Boundary, LaneReferenceGeometry, and LaneBundleReferenceGeometry are defined as subclasses for LaneLevelGeomerty. The LaneLevelGeometry is a sub-entity of the LineSegment entity that connects a sequence of Nodes. A Node represents a point in the three-dimensional space, and its attributes include its latitude, longitude, and altitude and it is the fundamental entity of the data model.

A landmark is located by a single Node and the Node serves as the center point for the landmark. A landmark is bounded by its Boundary. The boundary sketches the shape of the vertical view of the landmark and it could be an arbitrary polygon. We propose to make photo as one of the attributes of the Landmark. In cooperation with the cameras

and sensors on autonomous vehicles, photos taken beforehand can play an important role in applications such as vehicle self-localization, traffic sign recognition, and traffic light recognition. The On-roadLandmarks are associated with the Lanes and LaneBundles. Example of On-roadLandmarks is the speed sign. A speed sign is represented by a 3D point cloud that depicts the shape and the size of the object in the real world. High-resolution photos of the sign are stored to facilitate the recognition process of the sensors to save computation power and increase safety. The absolute location of the sign is represented by a Node and its Boundary. The sign is associated with the LaneBundle and the range attribute that it has an effect on. The Outdoor Landmarks are not associated with Lanes or LaneBundles because they have no impact on the attributes. One of the examples could be a tree. The Node and the Boundary for the tree are located away from the LaneBundle representing the road. The more detailed information, such as the profile of the tree and the 3D point cloud representing the shape of the tree can be registered. Although the Outdoor Landmarks are not directly associated with the road network, the information about the position and the shape of these landmarks can be used for localization and incident avoidance.

6.4 Discussion

This section provides further discussion on three main points: (1) compatibility of the proposed data model with existing data and applications, (2) potential simplification for the proposed data model, and (3) potential applications for HD maps other than self-driving technology.

Model compatibility. As the existing applications need to be supported along with the emerging applications, such as autonomous driving, our data model is compatible to serve all existing applications. Due to economic and infrastructural reasons, human drivers are expected to share the roads for at least a decade after a large number of fully autonomous cars are on roads. For example, autonomous cars are not expected to be affordable for many people at the beginning, and it will be neither affordable nor indispensable for cities to build new roads for autonomous cars isolating them from other cars. This is regardless of the fact that having a reasonable number of fully autonomous cars is not even the reality yet. Therefore, there is still a demand for human-machine interactions and the existing map applications are also essential for a long time ahead.

Our model maintains compatibility with the traditional applications through using the same core entities, Edge and Intersection, so any attribute or processing on the old edge and node entities can be performed on their corresponding entities. For Intersection entities, a center point of the intersection can be used to represent the whole intersection and treat it as the original node if needed. For Edge entities, a polyline that spans the different connected lane bundles can be used to represent the edge in the traditional format. Both the center point of an intersection and road segment polyline can be easily derived from the stored attributes in our model. This serves the existing applications with even improved potential features. For example, out of the most popular applications nowadays are navigation and route planning. Turn-by-turn navigation directions can be easily provided using our model treating edges and nodes as highlighted. Furthermore, the introduction of the lane level enables the navigation and routing applications to give more detailed instruc-

tions, which help to further improve the user experience for human drivers. Google Maps has started to use lane-level data to provide more detailed driving instructions to its users.

Model simplification. In Section 6.3, we described how we extend the existing Intersection-Edge format to model the HD map data. Reflecting on this extension while assuming we do not have the constraint of compatibility with the traditional node-edge model might lead to a potential simplification for the proposed model that worth further explorations. This simplification makes use of the observation that Edge and Intersection entities, which are marked with the dashed rectangle in Figure 6.3, are mere containers for a set of lane bundles. In fact, the definition of both entities has totally replaced with a set of lane bundles that have different spatial relationships. In the Edge, the set of lane bundles form a connected sequence, where a lane bundle starts wherever the previous one ends. In the Intersection, the set of lane bundles are geographically overlapping and connects different Edge instances. In all cases, each lane bundle encapsulates its own geometry that defines its geographical location and orientation. The entities Edge and Intersection are mere logical grouping that do not hold any attributes in the proposed model. Some popular applications, such as routing, might not need this logical grouping and can sufficiently use the lane bundles to provide the functionality. In this case, we can remove the abstraction of Edge and Intersection entities and use sets of LaneBundles as replacements for them. It worth exploration if this is applicable for the vast majority of applications, or there is still a significant need to keep the Edge and Intersection entities.

Before discussing the implications of the suggested simplification on other entities in the data model, it is essential to highlight merits and potential drawbacks of such sim-

plification. One of the obvious drawbacks is the lack of compatibility with existing map models and applications. In other words, this simplification will convert the model into a model that purely serve HD map application and will make integration with traditional mapping application harder. In specific, it will be a necessity to map traditional edges to multiple lane bundles, that are not currently grouped at the logical level, and the same for traditional intersections. So, using this in applications need to be conditioned with an awareness of such limitation. On the other hand, the simplification will make the model simpler and will ensure more integrity for the underlying data due to removing one layer of redundancy.

Removing the Edge and Intersection logical grouping will have minimal effect on the rest of the modeled entities. First, it will not cause the model to collapse. The Intersection and Edges record the fundamental topology of the road networks. Since the Intersections and Edges are composed of lane bundles, the topology information is maintained by the relative position of different lane bundles. Second, it is worth considering how to accommodate the attributes associated with the Edges and Intersections. It is straightforward to map the Edge oriented attributes on to the LaneBundleReferenceGeometry. Since the Edge is decomposed into a sequence of LaneBundles that are not intersected, the attributes are also distributed into segments. The attributes associated with the intersections will be mapped to the specific lanes or lane bundles. For example, the Intersection traversal information will be reflected by whether there are turning lanes for the corresponding direction. The traffic lights will be associated with the boundary of the corresponding lanes to put a condition about when the vehicles can pass a certain boundary to proceed. Overall,

all attributes that can be associated with both Edge and Intersection entities can be also associated with one of the other entities.

Potential applications on HD maps. Although high-definition maps are mainly motivated by self-driving cars, this kind of data can be used in endless applications that could make use of such fine-granularity information in other contexts. For example, an on-going research in forestry makes use of Google street images to identify types of trees that suffer from certain problems in Vancouver, Canada, and researchers are willing to expand their techniques to other geographical areas in North America and Europe. Another example is a tree atlas for Southern California that is being developed based on Google Earth images data that document forests and trees in the region. A third potential example is studying the development of urban areas and human migrations through analyzing data from different time snapshots. In nutshell, the on-going efforts to build high-definition maps will help researchers from different disciplines to study different problems with a relatively cheap source of data compared to existing methods.

We are also aware that one of the major features of the HD maps is the enormous map data size due to the details added. Currently, there are also room for improvement for efficient data management [168] and format compactness and efficiency [176]. These shortcomings may prevent the HD map from replacing the existing digital maps, but we have a positive prospect about with the development of communication technology and data access technology.

6.5 Conclusions

To cater to the requirement for highly detailed and accurate maps by self-driving cars, HD map has been widely discussed and there are several commercial HD map data models from different map providers. However, there are arguments that the existing HD map data models are not research-friendly. In this chapter, we propose a standard research-friendly HD map data schema by extending the existing node-edge structure in the tradition map data models. The data model includes two major parts, road network, and landmarks. The chapter also discussed the compatibility of the model with existing applications, potential simplifications to the model, and further applications for HD maps beyond self-driving cars. The data model serves as a generic framework for a more detailed design. Meanwhile, the management [177] and the efficient update of such a data model are also examples for future research directions.

Chapter 7

Conclusions

This dissertation encapsulates our endeavors to enhance the scalability and expressiveness of spatial analysis algorithms.

In the outset, we introduce *Pyneapple*, an open-source Python library geared toward scalable and expressive spatial data analysis. The release of *Pyneapple* extends our research contributions in regionalization, machine learning, and group-by aggregation techniques to the broader social science community. The library offers intuitive user interaction through a graphical interface that allows for attribute selection and constraint threshold adjustments via sliding bars. APIs are provided in both Python and Java environments, facilitating usage across a diverse set of developers and researchers. Notably, our experience in integrating low-level Java APIs with Python showcases potential avenues for bridging the gap between algorithm designers and social scientists more seamlessly. The integration of the GPT-3.5 model into our web application has yielded noteworthy results. This advanced AI model not only translates natural language queries into formal database requests but

also autonomously interprets data columns, obviating manual adjustments. This successful adaptation illustrates the viability of cutting-edge AI models as intermediaries between users and core algorithms, as well as between different software components.

In terms of algorithms within the *Pyneapple* library, we focus on our innovative work on expressive max-p regionalization (EMP) and spatial hotspot detection on spatial networks. EMP extends the conventional max-p regions problem by incorporating multiple constraints, aggregation functions, and range operators, thereby elevating its expressiveness significantly. To address EMP, we propose *FaCT*, a three-phase heuristic algorithm. *FaCT* integrates feasibility checks, constraint validation, and efficient construction phases, allowing it to tackle larger and more complex datasets compared to existing methods.

Additionally, the dissertation improves the scalability of spatial network hotspot detection through sophisticated graph traversal algorithms, which is also an indepensible component of the *Pyneapple* library. Utilizing edge-level approximations, our methods achieve near-perfect accuracy while drastically reducing computation time, as evidenced by a five-order-of-magnitude improvement in speed compared to existing solutions. In future works, we would like explore the possibility of utilizing the machine learning, especially deep learning technologies to further improve the scalability of hotspot detection on spatial networks. Because the spatial networks are in nature graphs that caries spatial properties, it would fit into the scheme of graph deep learning, such as graph neural networks, graph convolution networks, and graph attention networks.

In Chapter 5, we present *CleanUpOurWorld*, a tool designed for scalable visualization of global anthropogenic litter data. The platform leverages machine learning and

natural language processing to manage, clean, and visualize over 420,000 data points. The tool's design reflects the increasing utility and necessity of advanced algorithms in handling diverse types of spatial data.

Lastly, Chapter 6 introduces *HiDaM*, a high-definition map data model geared toward research applications. *HiDaM* is purposefully expressive, supporting a wide range of applications by including fine-grained road and environmental data. Through its high-level abstractions, *HiDaM* maintains compatibility with existing GIS applications. As the field of high-definition mapping remains an active area of research, we anticipate that our contributions will stimulate further investigations and developments in this burgeoning discipline.

In summary, this dissertation makes substantial contributions to advancing the fields of scalability and expressiveness within spatial analysis. Our work encompasses a range of activities, from data discovery and analysis to visualization and modeling. Specifically, we've developed robust algorithms, intuitive platforms, and data models that not only increase the efficiency of processing large datasets but also enhance the capacity to capture intricate spatial relationships and constraints. These advancements enable more comprehensive and nuanced insights into geographical patterns, thereby enriching the capabilities of researchers and practitioners in social science and other disciplines that utilize spatial data analysis techniques. Looking ahead, we are enthusiastic about leveraging our acquired knowledge and insights to further expand the frontiers of spatial analysis. Our aim is to continue producing research that is both innovative and practically applicable, with a focus on addressing real-world challenges and influencing policy through enhanced spatial understanding.

Bibliography

[1] John R Logan. Making a place for space: Spatial thinking in social science. *Annual review of sociology*, 38:507–524, 2012.

[2] David Darmofal. *Spatial analysis for the social sciences*. Cambridge University Press, 2015.

[3] Luc Anselin. The future of spatial analysis in the social sciences. *Geographic information sciences*, 5(2):67–76, 1999.

[4] John Snow. On the mode of communication of cholera. *Edinburgh medical journal*, 1(7):668, 1856.

[5] Ivan Franch-Pardo, Brian M Napoletano, Fernando Rosete-Verges, and Lawal Billa. Spatial analysis and gis in the study of covid-19. a review. *Science of the total environment*, 739:140033, 2020.

[6] Roberto Benedetti, Federica Piersimoni, Giacomo Pignataro, and Francesco Vidoli. The Identification of Spatially Constrained Homogeneous Clusters of Covid-19 Transmission in Italy. *RSPP*, 12:1169–1187, 2020.

[7] Dirk U Pfeiffer, Timothy P Robinson, Mark Stevenson, Kim B Stevens, David J Rogers, and Archie CA Clements. *Spatial analysis in epidemiology*. OUP Oxford, 2008.

[8] Anthony GO Yeh. Urban planning and gis. *Geographical information systems*, 2(877-888):1, 1999.

[9] Antonio Páez and Darren M Scott. Spatial statistics for urban analysis: A review of techniques with examples. *GeoJournal*, 61:53–67, 2004.

[10] MK Jat, Pradeep Kumar Garg, and Deepak Khare. Modelling of urban growth using spatial analysis techniques: a case study of ajmer city (india). *International Journal of Remote Sensing*, 29(2):543–567, 2008.

[11] Harvey J Miller. Potential contributions of spatial analysis to geographic information systems for transportation (gis-t). *Geographical Analysis*, 31(4):373–399, 1999.

[12] Masanobu Kii, Hitomi Nakanishi, Kazuki Nakamura, and Kenji Doi. Transportation and spatial development: An overview and a future direction. *Transport Policy*, 49:148–158, 2016.

[13] Rakibul Ahasan and Md Mahbub Hossain. Leveraging gis and spatial analysis for informed decision-making in covid-19 pandemic. *Health policy and technology*, 10(1):7, 2021.

[14] Terry W Griffin, Craig L Dobbins, Tony J Vyn, Raymond JGM Florax, and James M Lowenberg-DeBoer. Spatial analysis of yield monitor data: Case studies of on-farm trials and farm management decision making. *Precision Agriculture*, 9:269–283, 2008.

[15] Yifei Xue and Donald E Brown. Spatial analysis with preference specification of latent decision makers for criminal event prediction. *Decision support systems*, 41(3):560–573, 2006.

[16] Natasha Noy. Discovering millions of datasets on the web. `https://blog.google/products/search/discovering-millions-datasets-web/`, Jan 2020.

[17] Natasha Noy. When the web is your data lake: Creating a search engine for datasets on the web. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 801–801, 2020.

[18] Luc Anselin, Ibnu Syabri, and Youngihn Kho. Geoda: an introduction to spatial data analysis. In *Handbook of applied spatial analysis: Software tools, methods and applications*, pages 73–89. Springer, 2009.

[19] Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

[20] The White House. Circular no. a-16 revised. `https://obamawhitehouse.archives.gov/omb/circulars`$_{\text{a}}$`016`$_{\text{r}}$`ev/#2`.

[21] Peter A Burrough, Rachael A McDonnell, and Christopher D Lloyd. *Principles of geographical information systems*. Oxford University Press, USA, 2015.

[22] Ralf Hartmut Güting. An introduction to spatial database systems. *the VLDB Journal*, 3:357–399, 1994.

[23] Zenon Pylyshyn. The role of location indexes in spatial perception: A sketch of the finst spatial-index model. *Cognition*, 32(1):65–97, 1989.

[24] Stewart Fotheringham and Peter Rogerson. *Spatial analysis and GIS*. Crc Press, 2013.

[25] Shashi Shekhar. *Spatial databases*. Pearson Education India, 2007.

[26] Philippe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial databases: with application to GIS*. Morgan Kaufmann, 2002.

[27] Luc Anselin. The future of spatial analysis in the social sciences. *Geographic information sciences*, 5(2):67–76, 1999.

[28] Michael F Goodchild, Luc Anselin, Richard P Appelbaum, and Barbara Herr Harthorn. Toward spatially integrated social science. *International Regional Science Review*, 23(2):139–159, 2000.

[29] Luc Anselin. Spatial@uchicago.

[30] Sergio J Rey and Luc Anselin. Pysal: A python library of spatial analytical methods. In *Handbook of applied spatial analysis: Software tools, methods and applications*, pages 175–193. Springer, 2009.

[31] Lauren M Scott and Mark V Janikas. Spatial statistics in arcgis. In *Handbook of applied spatial analysis: Software tools, methods and applications*, pages 27–41. Springer, 2009.

[32] Biden-Harris Administration launches version 1.0 of climate and economic justice screening tool, key step in implementing president Biden's Justice40 Initiative. `https://www.whitehouse.gov/ceq/news-updates/2022/11/22/biden-harris-administration-launches-version-1-0-of-climate-and-economic-justice-screening-tool-key-step-in-implementing-president-bidens-justice40-initiative/`, Nov 2022.

[33] Rui Fragoso, Conceição Rego, and Vladimir Bushenkov. Clustering of territorial areas: A multi-criteria districting problem. *Journal of Quantitative Economics*, 14(2):179–198, 2016.

[34] Sergio J Rey and Myrna L Sastré-Gutiérrez. Interregional Inequality Dynamics in Mexico. *SEA*, 5:277–298, 2010.

[35] Miguel Camacho-Collados, Federico Liberatore, and José Miguel Angulo. A multi-criteria police districting problem for the efficient and effective design of patrol sector. *European journal of operational research*, 246(2):674–684, 2015.

[36] Daniel Arribas-Bel and Charles R Schmidt. Self-Organizing Maps and the US Urban Spatial Structure. *EPB*, 40:362–371, 2013.

[37] Juan C Duque, Jorge E Patino, Luis A Ruiz, and Josep E Pardo-Pascual. Measuring Intra-urban Poverty Using Land Cover and Texture Metrics Derived from Remote Sensing Data. *LUP*, 135:11–21, 2015.

[38] Jorge E Patino, Juan C Duque, Josep E Pardo-Pascual, and Luis A Ruiz. Using Remote Sensing to Assess the Relationship Between Crime and the Urban Layout. *Applied Geography*, 55:48–60, 2014.

[39] Seth E Spielman and David C Folch. Reducing uncertainty in the american community survey through data-driven regionalization. *PloS one*, 10(2):e0115626, 2015.

[40] Douglas A Stow, Christopher D Lippitt, and John R Weeks. Geographic Object-based Delineation of Neighborhoods of Accra, Ghana using QuickBird Satellite Imagery. *PE&RS*, 76:907–914, 2010.

[41] Yongyi Liu, Ahmed R Mahmood, Amr Magdy, and Sergio Rey. Pruc: P-regions with user-defined constraint. *Proceedings of the VLDB Endowment*, 15(3):491–503, 2021.

[42] Hussah Alrashid, Yongyi Liu, and Amr Magdy. Smp: scalable max-p regionalization. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, pages 1–4, 2022.

[43] Yunfan Kang and Amr Magdy. Emp: Max-p regionalization with enriched constraints. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1914–1926. IEEE, 2022.

[44] Ate Poorthuis. How to draw a neighborhood? the potential of big data, regionalization, and community detection for understanding the heterogeneous nature of urban neighborhoods. *Geographical Analysis*, 50(2):182–203, 2018.

[45] Ran Wei, Sergio Rey, and Elijah Knaap. Efficient regionalization for spatially explicit neighborhood delineation. *International Journal of Geographical Information Science*, 35(1):135–151, 2021.

[46] Mohammad Reza Shahneh, Samet Oymak, and Amr Magdy. A-gwr: Fast and accurate geospatial inference via augmented geographically weighted regression. In *Proceedings of the 29th international conference on advances in geographic information systems*, pages 564–575, 2021.

[47] Yongyi Liu, Yunfan Kang, Amr Magdy, and Ahmed Mahmood. Scalable Local K-function on Spatial Networks. In *Under submission to ACM SIGSPATIAL*, 2023.

[48] Brian D Ripley. The Second-order Analysis of Stationary Point Processes. *Journal of Applied Probability*, 13(2):255–266, 1976.

[49] Atsuyuki Okabe, Kei-ichi Okunuki, and Shino Shiode. Sanet: a toolbox for spatial analysis on a network. *Geographical analysis*, 38(1):57–66, 2006.

[50] Suman Rakshit, Adrian Baddeley, and Gopalan Nair. Efficient code for second order analysis of events on a linear network. *Journal of Statistical Software*, 90:1–37, 2019.

[51] Laila Abdelhafeez, Amr Magdy, and Vassilis J Tsotras. Scalable spatial groupby aggregations over complex polygons. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 449–452, 2020.

[52] Laila Abdelhafeez, Amr Magdy, and Vassilis J Tsotras. Sgpac: generalized scalable spatial groupby aggregations over complex polygons. *GeoInformatica*, pages 1–28, 2023.

[53] Laila Abdelhafeez. Scalable spatial queries in big data systems. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, pages 328–330. IEEE, 2022.

[54] MagdyLab. Pyneapple notebooks. `https://github.com/MagdyLab/Pyneapple/tree/main/notebooks`.

[55] Marc P Armstrong, Gerard Rushton, Rex Honey, Brian T Dalziel, Panos Lolonis, Suranjan De, and Paul J Densham. Decision Support for Regionalization: A Spatial Decision Support System for Regionalizing Service Delivery Systems. *CEUS*, 15:37–53, 1991.

[56] Kendra Spence Cheruvelil, Patricia A Soranno, Mary T Bremigan, Tyler Wagner, and Sherry L Martin. Grouping Lakes for Water Quality Assessment and Monitoring: The Roles of Regionalization and Spatial Scale. *JEM*, 41:425–440, 2008.

[57] Ahmed El Kenawy, Juan I López-Moreno, and Sergio M Vicente-Serrano. Summer Temperature Extremes in Northeastern Spain: Spatial Regionalization and Links to Atmospheric Circulation (1960–2006). *TAC*, 113:387–405, 2013.

[58] Sarah Schönbrodt-Stitt, Anna Bosch, Thorsten Behrens, Heike Hartmann, Xuezheng Shi, and Thomas Scholten. Approximation and Spatial Regionalization of Rainfall Erosivity Based on Sparse Data in a Mountainous Catchment of the Yangtze River in Central China. *JESPR*, 20:6917–6933, 2013.

[59] Nina Bullen, Graham Moon, and Kelvyn Jones. Defining localities for health planning: a gis approach. *Social Science & Medicine*, 42(6):801–816, 1996.

[60] Zhao Chen, Peng Cheng, Lei Chen, Xuemin Lin, and Cyrus Shahabi. Fair task assignment in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, 13(12):2479–2492, 2020.

[61] Tianshu Song, Yongxin Tong, Libin Wang, Jieying She, Bin Yao, Lei Chen, and Ke Xu. Trichromatic online matching in real-time spatial crowdsourcing. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1009–1020. IEEE, 2017.

[62] David J Rossiter and Ronald J Johnston. Program group: the identification of all possible solutions to a constituency-delimitation problem. *Environment and Planning A*, 13(2):231–238, 1981.

[63] Juan C Duque, Luc Anselin, and Sergio J Rey. The max-p-regions problem. *Journal of Regional Science*, 52(3):397–419, 2012.

[64] Seth E Spielman and David C Folch. Reducing Uncertainty in the American Community Survey through Data-driven Regionalization. *PloS ONE*, 10:e0115626, 2015.

[65] Yunfan Kang. Modeling the enriched Max-P region problem asa Mixed-integer programming problem. `https://github.com/YunfanKang/FaCT/blob/main/EMP-MIP/EMP`$_{\text{M}}$`IP`$_{\text{F}}$`ormulation.pdf`, 2022.

[66] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[67] Yunfan Kang. Solving the MIP formulation of EMP with Gurobi solver. `https://github.com/YunfanKang/FaCT/blob/main/EMP-MIP/`, 2022.

[68] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[69] Anuška Ferligoj and Vladimir Batagelj. Clustering with relational constraint. *Psychometrika*, 47(4):413–426, 1982.

[70] Jan Byfuglien and Asbjørn Nordgård. Region-building—a comparison of methods. *Norwegian Journal of Geography*, 1973.

[71] Leonard P Lefkovitch. Conditional clustering. *Biometrics*, pages 43–58, 1980.

[72] Fionn Murtagh. A survey of algorithms for contiguity-constrained clustering and related problems. *The computer journal*, 28(1):82–88, 1985.

[73] AD Gordon. A survey of constrained classification. *Computational Statistics & Data Analysis*, 21(1):17–29, 1996.

[74] Pierre Hansen, Brigitte Jaumard, Christophe Meyer, Bruno Simeone, and Valeria Doring. Maximum split clustering under connectivity constraints. *Journal of Classification*, 20(2):143–180, 2003.

[75] Juan C Duque, Richard L Church, and Richard S Middleton. The p-regions problem. *Geographical Analysis*, 43(1):104–126, 2011.

[76] Hyun Kim, Yongwan Chun, and Kamyoung Kim. Delimitation of Functional Regions Using a P-Regions Problem Approach. *IRSR*, 38:235–263, 2015.

[77] Kamyoung Kim, Yongwan Chun, and Hyun Kim. p-Functional Clusters Location Problem for Detecting Spatial Clusters with Covering Approach. *Geographical Analysis*, 49:101–121, 2017.

[78] Xinyue Ye, Bing She, and Samuel Benya. Exploring Regionalization in the Network Urban Space. *JGSA*, 2:4, 2018.

[79] Jason Laura, Wenwen Li, Sergio J Rey, and Luc Anselin. Parallelization of a Regionalization Heuristic in Distributed Computing Platforms–a Case Study of Parallel-P-Compact-Regions Problem. *IJGIS*, 29:536–555, 2015.

[80] Wenwen Li, Richard L Church, and Michael F Goodchild. An Extendable Heuristic Framework to Solve the P-Compact-Regions Problem for Urban Economic Modeling. *CEUS*, 43:1–13, 2014.

[81] Wenwen Li, Richard L Church, and Michael F Goodchild. The p-Compact-Regions Problem. *Geographical Analysis*, 46:250–273, 2014.

[82] David C Folch and Seth E Spielman. Identifying regions based on flexible user-defined constraints. *International Journal of Geographical Information Science*, 28(1):164–184, 2014.

[83] Bing She, Juan C Duque, and Xinyue Ye. The network-max-p-regions model. *IJGIS*, 31:962–981, 2017.

[84] Jean Homian Danumah, Samuel Nii Odai, Bachir Mahaman Saley, Joerg Szarzynski, Michael Thiel, Adjei Kwaku, Fernand Koffi Kouame, and Lucette You Akpa. Flood risk assessment and mapping in abidjan district using multi-criteria analysis (ahp) model and geoinformation techniques,(cote d'ivoire). *Geoenvironmental Disasters*, 3(1):1–13, 2016.

[85] Tatjana Vilutienė and Edmundas Kazimieras Zavadskas. The application of multi-criteria analysis to decision support for the facility management of a residential district. *Journal of Civil Engineering and Management*, 9(4):241–252, 2003.

[86] Seda Yanık, Joerg Kalcsics, Stefan Nickel, and Burcin Bozkaya. A multi-period multi-criteria districting problem applied to primary care scheme with gradual assignment. *International Transactions in Operational Research*, 26(5):1676–1697, 2019.

[87] Stan Openshaw. A regionalisation program for large data sets. *Computer Applications*, 3(4):136–147, 1973.

[88] Stan Openshaw. Classifying and regionalizing census data. *Census users' handbook*, pages 239–270, 1995.

[89] Renato M Assunção, Marcos Corrêa Neves, Gilberto Câmara, and Corina da Costa Freitas. Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees. *International Journal of Geographical Information Science*, 20(7):797–811, 2006.

[90] Robert S Garfinkel and George L Nemhauser. Optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B–495, 1970.

[91] Stan Openshaw. A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Transactions of the institute of british geographers*, pages 459–472, 1977.

[92] Orhun Aydin, Mark V Janikas, Renato Assunção, and Ting-Hwan Lee. Skatercon: Unsupervised regionalization via stochastic tree partitioning within a consensus framework using random spanning trees. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pages 33–42, 2018.

[93] Konstantin Andreev and Harald Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.

[94] NP-hardness of MP-regions. `https://cs.ucr.edu/~amr/MP_Regionalization_NPHardness.pdf`, 2021.

[95] Ran Wei, Sergio Rey, and Elijah Knaap. Efficient Regionalization for Spatially Explicit Neighborhood Delineation. *IJGIS*, 35:1–17, 2020.

[96] Yunfan Kang. Fact. `https://github.com/YunfanKang/FaCT`, 2021.

[97] Southern California Association of Governments - SCAG. `https://scag.ca.gov/`, 2020. May 2020.

[98] U.S. Census Bureau. Explore census data. `https://data.census.gov/cedsci/`.

[99] Southern California Association of Governments. Scag open data portal. `https://gisdata-scag.opendata.arcgis.com/`.

[100] Welcome to the QGIS project! `https://www.qgis.org/`, 2020. May 2020.

[101] Wenquan Li, Qinma Kang, Hanzhang Kong, Chao Liu, and Yunfan Kang. A novel iterated greedy algorithm for detecting communities in complex network. *Social Network Analysis and Mining*, 10:1–17, 2020.

[102] Qinma Kang, Hong He, and Jun Wei. An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of parallel and distributed computing*, 73(8):1106–1115, 2013.

[103] Qinma Kang, Hong He, and Huimin Song. Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm. *Journal of Systems and Software*, 84(6):985–992, 2011.

[104] Svante Berglund and Anders Karlström. Identifying Local Spatial Association in Flow Data. *Journal of Geographical Systems*, 1:219–236, 1999.

[105] Atsuyuki Okabe and Kokichi Sugihara. *Spatial Analysis Along Networks: Statistical and Computational Methods*. John Wiley & Sons, 2012.

[106] Yiqun Xie, Shashi Shekhar, and Yan Li. Statistically-robust Clustering Techniques for Mapping Spatial Hotspots: A Survey. *ACM Computing Surveys (CSUR)*, 55(2):1–38, 2022.

[107] Shoaib Khalid, Fariha Shoaib, Tianlu Qian, Yikang Rui, Arezu Imran Bari, Muhammad Sajjad, Muhammad Shakeel, and Jiechen Wang. Network Constrained Spatio-temporal Hotspot Mapping of Crimes in Faisalabad. *Applied Spatial Analysis and Policy*, 11:599–622, 2018.

[108] Xun Tang, Emre Eftelioglu, and Shashi Shekhar. Detecting Isodistance Hotspots on Spatial Networks: A Summary of Results. In *International Symposium on Spatial and Temporal Databases*, pages 281–299, 2017.

[109] Tsz Nam Chan, Leong Hou U, Yun Peng, Byron Choi, and Jianliang Xu. Fast Network k-function-based Spatial Analysis. *Proceedings of the VLDB Endowment*, 15(11):2853–2866, 2022.

[110] Benjamin Romano and Zhe Jiang. Visualizing Traffic Accident Hotspots based on Spatial-temporal Network Kernel Density Estimation. In *Proceedings of the International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*, pages 1–4, 2017.

[111] Yan Shi, Yuanfang Chen, Min Deng, Liang Xu, and Jiaqin Xia. Discovering Source Areas of Disease Outbreaks based on Ring-shaped Hotspot Detection in Road Network Space. *International Journal of Geographical Information Science*, 36(7):1343–1363, 2022.

[112] Tianfu Wang, Chang Ren, Yun Luo, and Jing Tian. NS-DBSCAN: A Density-based Clustering Algorithm in Network Space. *ISPRS International Journal of Geo-Information*, 8(5):218, 2019.

[113] Thérèse Steenberghen, Koen Aerts, and Isabelle Thomas. Spatial Clustering of Events on a Network. *Journal of Transport Geography*, 18(3):411–418, 2010.

[114] Yihui Ren, Mária Ercsey-Ravasz, Pu Wang, Marta C González, and Zoltán Toroczkai. Predicting Commuter Flows in Spatial Networks Using a Radiation Model Based on Temporal Ranges. *Nature Communications*, 5(1):5347, 2014.

[115] Manas Joshi, Arshdeep Singh, Sayan Ranu, Amitabha Bagchi, Priyank Karia, and Puneet Kala. FoodMatch: Batching and Matching for Food Delivery in Dynamic Road Networks. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(1):1–25, 2022.

[116] Marilena Kampa and Elias Castanas. Human Health Effects of Air Pollution. *Environmental pollution*, 151(2):362–367, 2008.

[117] Xun Tang, Emre Eftelioglu, Dev Oliver, and Shashi Shekhar. Significant Linear Hotspot Discovery. *IEEE Transactions on Big Data*, 3(2):140–153, 2017.

[118] Xun Tang, Jayant Gupta, and Shashi Shekhar. Linear Hotspot Discovery on All Simple Paths: A Summary of Results. In *Proceedings of the International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*, pages 476–479, 2019.

[119] 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.

[120] Min Deng, Xuexi Yang, Yan Shi, Jianya Gong, Yang Liu, and Huimin Liu. A Density-based Approach for Detecting Network-constrained Clusters in Spatial Point Events. *International Journal of Geographical Information Science*, 33(3):466–488, 2019.

[121] Man Lung Yiu and Nikos Mamoulis. Clustering Objects on a Spatial Network. In *Proceedings of the International Conference on Management of Data (ACM SIGMOD)*, pages 443–454, 2004.

[122] Shino Shiode and Narushige Shiode. Detection of Multi-scale Clusters in Network Space. *International Journal of Geographical Information Science*, 23(1):75–92, 2009.

[123] Tsz Nam Chan, Zhe Li, Leong Hou U, Jianliang Xu, and Reynold Cheng. Fast Augmentation Algorithms for Network Kernel Density Visualization. *Proceedings of the VLDB Endowment*, 14(9):1503–1516, 2021.

[124] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 2016.

[125] Ikuho Yamada and Jean-Claude Thill. Local Indicators of Network-constrained Clusters in Spatial Point Patterns. *Geographical Analysis*, 39(3):268–292, 2007.

[126] Chicago Data Portal. Chicago Crime Data. `https://data.cityofchicago.org/`, 2023.

[127] Yan Li, Yiqun Xie, Pengyue Wang, Shashi Shekhar, and William Northrop. Significant Lagrangian Linear Hotspot Discovery. In *Proceedings of the 13th*

ACM SIGSPATIAL International Workshop on Computational Transportation Science, pages 1–10, 2020.

[128] Sara Mclafferty. Disease Cluster Detection Methods: Recent Developments and Public Health Implications. *Annals of GIS*, 21(2):127–133, 2015.

[129] Marc Souris. *Epidemiology and Geography: Principles, Methods and Tools of Spatial Analysis*. John Wiley & Sons, 2019.

[130] Ke Nie, Zhensheng Wang, Qingyun Du, Fu Ren, and Qin Tian. A Network-constrained Integrated Method for Detecting Spatial Cluster and Risk Location of Traffic Crash: A Case Study from Wuhan, China. *Sustainability*, 7(3):2662–2677, 2015.

[131] Martin A Andresen. *Environmental Criminology: Evolution, Theory, and Practice*. Routledge, 2019.

[132] Lisa Tompson, Henry Partridge, and Naomi Shepherd. Hot routes: Developing a New Technique for the Spatial Analysis of Crime. *Crime Mapping: A Journal of Research and Practice*, 1(1):77–96, 2009.

[133] Xiaohui Lin and Jianmin Xu. Road Network Partitioning Method Based on Canopy-kmeans Clustering Algorithm. *Archives of Transport*, 54(2):95–106, 2020.

[134] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[135] Arthur Getis and J Keith Ord. The Analysis of Spatial Association by Use of Distance Statistics. *Geographical Analysis*, 24(3):189–206, 1992.

[136] Atsuyuki Okabe, Kei-ichi Okunuki, and Shino Shiode. SANET: a Toolbox for Spatial Analysis on a Network. *Geographical Analysis*, 38(1):57–66, 2006.

[137] Edsger W Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[138] Waldo R Tobler. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, 46(sup1):234–240, 1970.

[139] Claude E Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[140] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*, volume 2. Springer, 2009.

[141] Kaggle. Seattle SDOT Collisions Data. https://www.kaggle.com/datasets/jonleon/seattle-sdot-collisions-data, 2020.

[142] NYCOpenData. New York City Motor Vehicle Collisions - Crashes. https://opendata.cityofnewyork.us, 2023.

[143] City of Detroit Open Data Portal. Detroit 911 Calls For Service. https://data.detroitmi.gov, 2023.

[144] Python OSMnx Library User reference. https://osmnx.readthedocs.io/en/stable/osmnx.html, 2020.

[145] Chenjuan Guo, Bin Yang, Ove Andersen, Christian S. Jensen, and Kristian Torp. EcoSky: Reducing Vehicular Environmental Impact Through Eco-routing. In *ICDE*, pages 1412–1415, 2015.

[146] Chenjuan Guo, Yu Ma, Bin Yang, Christian S. Jensen, and Manohar Kaul. EcoMark: Evaluating Models of Vehicular Environmental Impact. In *SIGSPATIAL*, pages 269–278, 2012.

[147] Impacts of Mismanaged Trash. https://www.epa.gov/trash-free-waters/impacts-mismanaged-trash, 2019.

[148] Marine Litter – A Growing Threat Worldwide. https://www.eea.europa.eu/highlights/marine-litter-2013-a-growing, 2017.

[149] Ocean Trash Isn't Just Bad For the Environment – It's Bad for Your State of Mind. https://www.washingtonpost.com/news/energy-environment/wp/2015/07/08/why-clean-beaches-are-so-important-if-you-want-a-relaxing-vacation/, 2017.

[150] Lawmakers reauthorize NOAA Marine Debris Program. https://www.americangeosciences.org/policy/news-brief/lawmakers-reauthorize-noaa-marine-debris-program, 2018.

[151] Smelly, Contaminated, Full of Disease: The World's Open Dumps Are Growing. https://www.theguardian.com/global-development/2014/oct/06/smelly-contaminated-disease-worlds-open-dumps, 2014.

[152] Waste Exposure and Skin Diseases. https://www.ncbi.nlm.nih.gov/pubmed/28209048, 2017.

[153] Plastics in the Ocean Affecting Human Health. https://serc.carleton.edu/NAGTWorkshops/health/case$_s$tudies/plastics.html, 2012.

[154] World Waste Platform. http://opendata.letsdoitworld.org, 2019.

[155] Let's do it! World. https://www.letsdoitworld.org/, 2018.

[156] Ocean Conservancy. https://oceanconservancy.org/, 2019.

[157] Litterati - A Litter Free World. https://www.litterati.org/, 2019.

[158] Marine Debris Tracker. http://www.marinedebris.engr.uga.edu/, 2019.

[159] Alice Ferguson Foundation. http://fergusonfoundation.org/trash-free-potomac-watershed-initiative/, 2019.

[160] US National Oceanic and Atmospheric Administration (NOAA). https://www.noaa.gov/, 2019.

[161] Farouk Ghallabi, Fawzi Nashashibi, Ghayath El-Haj-Shhade, and Marie-Anne Mittet. Lidar-based lane marking detection for vehicle positioning in an hd map. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2209–2214. IEEE, 2018.

[162] Jialin Jiao. Machine learning assisted high-definition map creation. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 367–373. IEEE, 2018.

[163] Peter Fischer, Seyed Majid Azimi, Robert Roschlaub, and Thomas Krauß. Towards hd maps from aerial imagery: Robust lane marking segmentation using country-scale imagery. *ISPRS International Journal of Geo-Information*, 7(12):458, 2018.

[164] Wonje Jang, Jhonghyun An, Sangyun Lee, Minho Cho, Myungki Sun, and Euntai Kim. Road lane semantic segmentation for high definition map. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1001–1006. IEEE, 2018.

[165] Kay Massow, Birgit Kwella, Niko Pfeifer, F Häusler, Jens Pontow, Ilja Radusch, Jochen Hipp, Frank Dölitzscher, and Martin Haueis. Deriving hd maps for highly automated driving from vehicular probe data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1745–1752. IEEE, 2016.

[166] Gellért Máttyus, Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Hd maps: Fine-grained road segmentation by parsing ground and aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3611–3619, 2016.

[167] Mahdi Javanmardi, Ehsan Javanmardi, Yanlei Gu, and Shunsuke Kamijo. Towards high-definition 3d urban mapping: Road feature-based registration of mobile mapping systems and aerial imagery. *Remote Sensing*, 9(10):975, 2017.

[168] Andi Zang, Xin Chen, and Goce Trajcevski. High definition maps in urban context. *SIGSPATIAL Special*, 10:15–20, 06 2018.

[169] Eijiro Takeuchi, Yuki Yoshihara, and Ninomiya Yoshiki. Blind area traffic prediction using high definition maps and lidar for safe driving assist. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2311–2316. IEEE, 2015.

[170] Manato Hirabayashi, Adi Sujiwo, Abraham Monrroy, Shinpei Kato, and Masato Edahiro. Traffic light recognition using high-definition map features. *Robotics and Autonomous Systems*, 111:62–72, 2019.

[171] Andi Zang, Zichen Li, David Doria, and Goce Trajcevski. Accurate vehicle self-localization in high definition map dataset. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles*, page 2. ACM, 2017.

[172] Wang Tao. Interdisciplinary urban gis for smart cities: advancements and opportunities. *Geo-spatial Information Science*, 16(1):25–34, 2013.

[173] GIS for Smart Cities. `https://www.esri.in/~/media/esri-india/files/pdfs/news/arcindianews/Vol9/gis-for-smart-cities.pdf`, 2015.

[174] A Gateway to Smart City Using 3D GIS Maps of Cities - LiDAR News Magazine. `http://www.lidarmag.com/PDF/LiDARNewsMagazine`$_{\text{G}}$`upta-SmartGISand3D`$_{\text{V}}$`ol5No4.pdf`, 2015.

[175] HD Map with RoadDNA info sheet. `http://download.tomtom.com/open/banners/HD`$_{\text{M}}$`ap`$_{\text{w}}$`ith`$_{\text{R}}$`oad`$_{\text{D}}$`NA`$_{\text{P}}$`roduct`$_{\text{I}}$`nfo`$_{\text{S}}$`heet.pdf`, 2018.

[176] NDS Open Lane Model. `http://www.openlanemodel.org`, 2019.

[177] Andi Zang, Shiyu Luo, Xin Chen, and Goce Trajcevski. Real-time applications using high resolution 3d objects in high definition maps. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2019.

[178] Rubbish and Diseases. `http://www.health.gov.au/internet/publications/publishing.nsf/Content/ohp-enhealth-manual-atsi-cnt-l~ohp-enhealth-manual-atsi-cnt-l-ch4~ohp-enhealth-manual-atsi-cnt-l-ch4.2`, 2010.

[179] Sven Bauer, Yasamin Alkhorshid, and Gerd Wanielik. Using high-definition maps for precise urban vehicle localization. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 492–497. IEEE, 2016.

[180] Franck Li, Philippe Bonnifait, and Javier Ibañez-Guzmán. Estimating localization uncertainty using multi-hypothesis map-matching on high-definition road maps. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[181] Shuran Zheng and Jinling Wang. High definition map-based vehicle localization for highly automated driving: Geometric analysis. In *2017 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–8. IEEE, 2017.

[182] Val L Mitchell. The regionalization of climate in the western united states. *Journal of Applied Meteorology*, 15(9):920–927, 1976.

[183] Robert Haining, Stephen Wise, and Jingsheng Ma. Designing and implementing software for spatial statistical analysis in a gis environment. *Journal of Geographical Systems*, 2(3):257–286, 2000.

[184] Stan Openshaw and Liang Rao. Algorithms for reengineering 1991 census geography. *Environment and planning A*, 27(3):425–446, 1995.

[185] Juan C Duque, Richard L Church, and Richard S Middleton. The P-Regions Problem. *Geographical Analysis*, 43:104–126, 2011.

[186] U.S. Census Bureau. TIGER/Line Shapefile, 2016, Series Information for the Current Census Tract State-based Shapefile, 2019. https://catalog.data.gov/dataset/tiger-line-shapefile-2016-series-information-for-the-current-census-tract-state-based-shapefile.

[187] V. Sindhu. Exploring Parallel Efficiency and Synergy for Max-P Region Problem Using Python. Master's thesis, Georgia State University, 2018.

[188] Juan Carlos Duque, Raúl Ramos, and Jordi Suriñach. Supervised Regionalization Methods: A Survey. *IRSR*, 30:195–220, 2007.

[189] Diansheng Guo. Regionalization with Dynamically Constrained Agglomerative Clustering and Partitioning (REDCAP). *IJGIS*, 22:801–823, 2008.

[190] Maurizio Maravalle and Bruno Simeone. A Spanning Tree Heuristic for Regional Clustering. *Communications in Statistics-theory and Methods*, 24:625–639, 1995.

[191] Diansheng Guo and Hu Wang. Automatic Region Building for Spatial Analysis. *Transactions in GIS*, 15:29–45, 2011.

[192] Kais Siala and Mohammad Youssef Mahfouz. Impact of the Choice of Regions on Energy System Models. *ESR*, 25:75–85, 2019.

[193] Sergio J Rey, Luc Anselin, David C Folch, Daniel Arribas-Bel, Myrna L Sastré Gutiérrez, and Lindsey Interlante. Measuring Spatial Dynamics in Metropolitan Areas. *EDQ*, 25:54–64, 2011.

[194] Fred Glover. Tabu Search—Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[195] C-N Fiechter. A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems. *DAM*, 51:243–267, 1994.

[196] Renato M Assunção, Marcos Corrêa Neves, Gilberto Câmara, and Corina da Costa Freitas. Efficient Regionalization Techniques for Socio-economic Geographical Units Using Minimum Spanning Trees. *IJGIS*, 20:797–811, 2006.

[197] Daniel Abadi, Anastasia Ailamaki, David Andersen, Peter Bailis, Magdalena Balazinska, Philip Bernstein, Peter Boncz, Surajit Chaudhuri, Alvin Cheung, AnHai Doan, et al. The Seattle Report on Database Research. *SIGMOD Record*, 48:44–53, 2020.

[198] Sergio J Rey and Luc Anselin. PySAL: A Python Library of Spatial Analytical Methods. In *SAGE*, pages 175–193. Springer, Heidelberg, Germany, 2010.

[199] Subhodip Biswas, Fanglan Chen, Zhiqian Chen, Andreea Sistrunk, Nathan Self, Chang-Tien Lu, and Naren Ramakrishnan. REGAL: A Regionalization Framework for School Boundaries. In *SIGSPATIAL*, pages 544–547, New York, NY, USA, 2019. Association for Computing Machinery.

[200] Subhodip Biswas, Fanglan Chen, Zhiqian Chen, Chang-Tien Lu, and Naren Ramakrishnan. Incorporating Domain Knowledge into Memetic Algorithms for Solving Spatial Optimization Problems. In *SIGSPATIAL*, pages 25–35, New York, NY, USA, 2020. Association for Computing Machinery.

[201] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A Model-based Approach to Attributed Graph Clustering. In *SIGMOD*, page 505–516, New York, NY, USA, 2012. Association for Computing Machinery.

[202] David Combe, Christine Largeron, Elöd Egyed-Zsigmond, and Mathias Géry. Combining Relations and Text in Scientific Network Clustering. In *ASONAM*, pages 1248–1253, Washington, D.C., USA, 2012. IEEE Computer Society.

[203] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph Clustering Based on Structural/Attribute Similarities. *PVLDB*, 2:718–729, 2009.

[204] Richard J Trudeau. *Introduction to Graph Theory*. Courier Corporation, 2013.

[205] Andrew B Lawson. Hotspot Detection and Clustering: Ways and Means. *Environmental and Ecological Statistics*, 17(2):231, 2010.

[206] Sergio J Rey, Luc Anselin, Xun Li, Robert Pahle, Jason Laura, Wenwen Li, and Julia Koschinsky. Open geospatial analytics with pysal. *ISPRS International Journal of Geo-Information*, 4(2):815–836, 2015.

Appendix for Local Hotspot Detection on Spatial Networks

.1 Complexity Analysis

This section presents the time and space complexity for all the proposed algorithms.

.1.1 Batch Based Traversal (BBT)

The time complexity of the BBT algorithm is $O(|E_e||V_l|\log|V_l|+|E_e|^2|V_e|\log|V_e|)$ and the space complexity is $O(|V_e|+|E_e|)$.

BBT has two steps. (1) *sorting the events along each edge*, which takes $O(|V_e|\log|V_e|)$. This is because it takes $O(|e|\log|e|)$ to sort the events along e. Consequently, the complexity of sorting events along all edges is

$$\sum_{e\in E_e} O(|e|\log|e|) \leq O(\log|V_e|) \sum_{e\in E_e} O(|e|) = O(|V_e|log|V_e|)$$

(2) *batch processing of events for every edge.* For an Edge $e = (P, Q)$, we first compute the location vertices reachable from P or Q within a predefined distance, which takes $O(|V_l|\log|V_l|+|E_e|)$. Then, for events along e, denote the complexity of evaluating the contribution of an edge $\tilde{e}$ to events along e as $C_{e,\tilde{e}}$. If $\tilde{e}$ falls into Category 1, then $C_{e,\tilde{e}} = O(1)$. Otherwise, $\tilde{e}$ falls into Category 3, and $C_{e,\tilde{e}} = O(|e|\log|\tilde{e}|)$. The reason is that BBT invokes binary search along $\tilde{e}$ twice for all events on edge e, which takes $O(|e|\log|\tilde{e}|)$. BBT visits $O(|E_e|)$ edges when evaluating events along e. The complexity of evaluating events

along e becomes

$$\sum_{\tilde{e} \in E_e} O(|e| \log |\tilde{e}|) \leq O(|V_e|) \sum_{\tilde{e} \in E_e} O(\log |\tilde{e}|) \leq O(|E_e||V_e| log |V_e|)$$

Consequently, batch processing events along e takes a complexity of $O(|V_l| \log |V_l| + |E_e||V_e| log |V_e|)$. Since there are $|E_e|$ edges, and the total time complexity for Step 2 for the batch processing of events along e becomes $O(|E_e||V_l| \log |V_l| + |E_e|^2 |V_e| log |V_e|)$

The total complexity for Step 2 is $O(|E_e||V_l| \log |V_l| + |E_e|^2 |V_e| log |V_e|)$. Combining the time complexities of Step 1 and Step 2, we conclude that the time complexity of BBT is $O(|E_e||V_l| \log |V_l| + |E_e|^2 |V_e| log |V_e|) + O(|V_e| log |V_e|) = O(|E_e||V_l| \log |V_l| + |E_e|^2 |V_e| log |V_e|)$.

The space complexity of BBT is $O(|V_e| + |E_e|)$. This is due to the storage required for each event vertex's distance to the endpoint of the edge it is located on, taking up $O(|V_e|)$ space. In addition, there are $|E_e|$ number of edges connecting different location vertices, which takes $O(|E_e|)$ space.

.1.2 Incremental Batched Traversal (IBT)

The time complexity of the IBT algorithm is $O(|E_e||V_l| \log |V_l| + |V_e||E_e| + |V_e| \log |V_e|)$ and the space complexity is $O(|V_e| + |E_e|)$. Similar to BBT, IBT also has two steps. (1) *sorting the events along each edge*, which takes $O(|V_e| \log |V_e|)$ as proved above, and (2) *batch processing of events in every edge*. However, IBT differs from BBT in Step 2, i.e., addressing edges in Category 3. In IBT, $C_{e,\tilde{e}}$ is bounded by $O(|e| + |\tilde{e}|)$ instead of $O(|e| \log |\tilde{e}|)$. The reason is that IBT utilizes two pointers,

$ptr_{\tilde{P}}$ and $ptr_{\tilde{Q}}$, and updates them in each iteration. Given the relationship between $sdist(P, \tilde{P})$, $sdist(Q, \tilde{P})$, and $w(e)$, three scenarios are considered for updating $ptr_{\tilde{P}}$:

Scenario 1: $sdist(P, \tilde{P}) + w(e) < sdist(Q, \tilde{P})$. Here, $r^i(\tilde{P})$ decreases monotonically, causing $ptr_{\tilde{P}}$ to move from $\tilde{Q}$ to $\tilde{P}$ as i increases, resulting $O(|\tilde{e}|)$ complexity.

Scenario 2: $sdist(Q, \tilde{P}) + w(e) < sdist(P, \tilde{P})$. The complexity is similar to Scenario 1, i.e., $O(|\tilde{e}|)$.

Scenario 3: $|sdist(P, \tilde{P}) - sdist(Q, \tilde{P})| \leq w(e)$. The update of pointer $ptr_{\tilde{P}}$ is monotonic for $i \in [1, j]$ and $i \in (j, |S_e|]$ but in opposite directions, where $j \in [1, |S_e|]$ such that for any events $S_e[i]$ with $i \leq j$, $sdist(S_e[i], P) \leq sdist(S_e[i], Q)$ and for any events $S_e[i]$ with $i > j$, $sdist(S_e[i], P) > sdist(S_e[i], Q)$. This also gives a complexity of $O(|\tilde{e}|)$.

Updating pointer $ptr_{\tilde{Q}}$ also takes $O(|\tilde{e}|)$, proven similarly. Consequently, processing events along e takes $O(|V_l|\log|V_l|+|V_e|)$, and the total complexity of Step 2 is $O(|E_e||V_l|\log|V_l|+|V_e||E_e|)$, since there are $|E_e|$ edges.

Hence, the overall time complexity of IBT is $O(|E_e||V_l|\log|V_l|+|V_e||E_e|+|V_e|\log|V_e|)$, and space complexity is $O(|V_e|+|E_e|)$, proved similarly to BBT.

.1.3 Approximate Hotspot Identification via Batched Edge Traversal (AH-IBT)

AH-IBT discovers hotspots without predefined radius by incrementally increasing the radius and computing the N and L using IBT for each radius. Similar to IBT, AH-IBT has two steps (1) sorting the events along each edge, and (2) batch processing of events and determine their optimal radii. Step 1 takes $O(|V_e|\log|V_e|)$, which is proved above.

Step 2 takes $O(|V_e||E_e|+|V_l|\log|V_l|)$ for the events along a single edge e. The reason is that it takes $O(|V_e|)$ to evaluate the statistical significance of events along e at a specific radius using IBT, and there can be $O(|E_e|)$ different radii. Plus the cost of incrementally retrieving location vertices closest to the endpoint of $\tilde{e}$, which takes $O(|V_l|\log|V_l|+|E_e|)$, the complexity of processing a single edge e is $O(|V_e||E_e|+|V_l|\log|V_l|)$. Taking into account all $|E_e|$ edges, the total complexity for Step 2 is $O(|V_e||E_e|^2+|E_e||V_l|\log|V_l|)$. Hence, the overall time complexity for AH-IBT is $O(|V_e||E_e|^2+|E_e||V_l|\log|V_l|+|V_e|\log|V_e|)$. The space complexity of AH-IBT is $O(|V_e|+|E_e|)$. The proof is similar to the above.

.2 Correctness Proof

In this section, we give the correctness proof for all the proposed algorithms. We prove that the proposed algorithms, i.e., GT, BBT, and IBT compute HDPR correctly by proving that they compute the N and L values correctly. Then, we prove that AH-IBT correctly answers HDWPR.

GT: For an event vertex $\tilde{\vartheta}$, if $sdist(\vartheta, \tilde{\vartheta}) \leq t$, $\tilde{\vartheta}$ is visited before GT's termination due to Dijkstra's algorithm properties. Any edge, or segment, $e = (u, v)$ contributing to ϑ's L value is considered when visiting u or v. This guarantees that all event and location vertices within t distance of ϑ are factored into N and L updates. GT terminates when all remaining vertices have distances to ϑ exceeding t.

BBT and IBT: For an event ϑ, BBT and IBT batch processes the computation of N and L with other events along the same edge e. Given another event, $\tilde{\vartheta}$ with $dist(\vartheta, \tilde{\vartheta}) \leq t$ attached to $\tilde{e} = (P, Q)$, $\tilde{\vartheta}$ is considered for updating N and L since $\tilde{e}$ fulfills either Category

1 or 3. BBT and IBT exclude any event $\overline{\vartheta}$ with $dist(\vartheta, \overline{\vartheta}) > t$ by pruning all edges that meet Category 2.

AH-IBT: The final radius t output by AH-IBT is indeed the radius that brings the first local maximum statistical significance. Suppose this local maximum is identified at iteration j in Algorithm 3 when expanding the radius for the event ϑ, then for all iterations of expanding the radius before j, the statistical significance of ϑ monotonically increases according to our definition in Section 4.5.2 . Then, from the j^{th} to $(j+1)^{th}$ iteration of of expanding the radius , the statistical significance of ϑ drops. Thus, the radius identified at at iteration j is the first local maximum statistical significance for ϑ.

.3 Additional Experiments

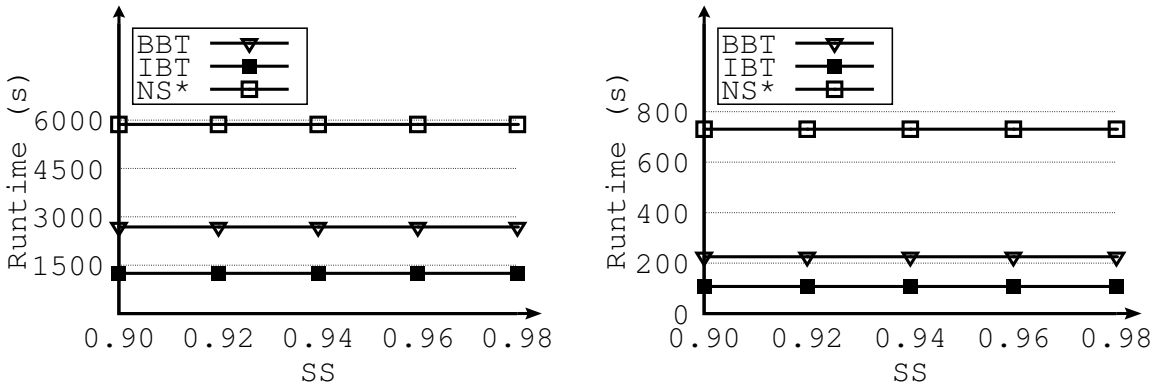
.3.1 Additional Experiments for HDPR

This section presents additional experiments for HDPR, i.e., hotspot detection with predefined radius.

The effect of statistical significance

Figure .1 shows the runtime of BBT, IBT, and NS* while changing the statistical significance. IBT consistently outperforms the other algorithms with up to 6.82 times speed up for the same reasons explained in Section 4.6.2. Also, Figure .1 shows that the runtime remains stable under different statistical significance levels. The reason is that changing the statistical significance required for hotspot detection does not significantly impact the computational overhead of N and L in all algorithms. However, changing the

required statistical significance only affects the final validation of hotspots, as discussed in Section 4.3. Although higher statistical significance results in fewer hotspots being qualified and detected, this has little impact on the overall runtime of algorithms.



(a) Chicago (b) Synth-Seattle-Random

Figure .1: Effect of Varying Statistical Significance (SS) in HDPR

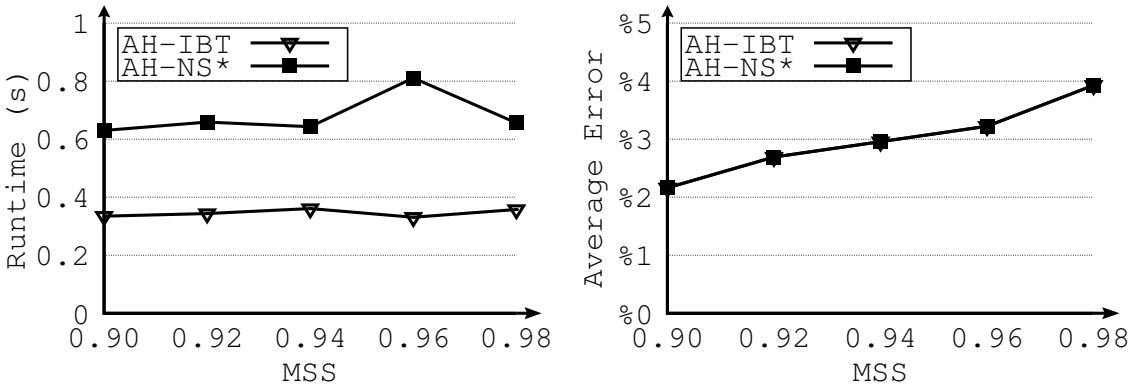
.3.2 Additional Experiments for HDWPR

This section presents additional experiments for HDWPR , i.e., hotspot detection without a predefined radius.

The effect of the minimum statistical significance

In this experiment, we investigate how the minimum statistical significance affects the runtime and average error for AH-IBT and AH-NS*. Notice that we do not consider NPP* in this experiment as NPP* does not have statistical significance as an algorithm parameter. Figure .2a shows that AH-IBT is up to 2.44 times faster than AH-NS*. The reason is that AH-IBT is based on IBT and AH-NS* is based on NS* which is slower than

IBT. Figure .2b shows that both AH-IBT and AH-NS* have the same average error because they adopt the same algorithm for identifying the radii of hotspots.



(a) Runtime (b) Accuracy

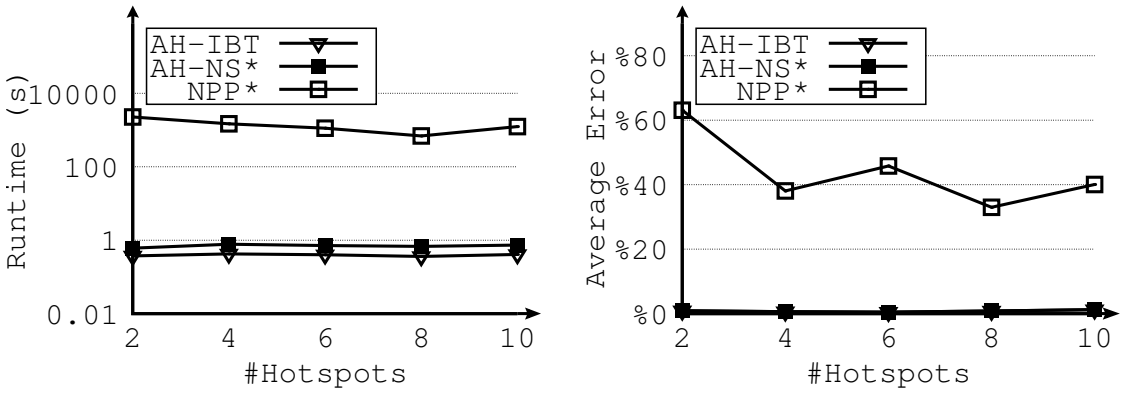
Figure .2: Effect of Varying Minimum Statistical Significance (MSS) in HDWPR

Although Figure .2b shows that the average error increases as MSS increases, there is no evident relationship between the average error and the minimum statistical significance. This is because under different MSS, the hotspots involved in computing the average error are different, which results in varying average error. However, in all cases, the average t error is less than 5%.

The effect of the number of hotspots and hotspots radii

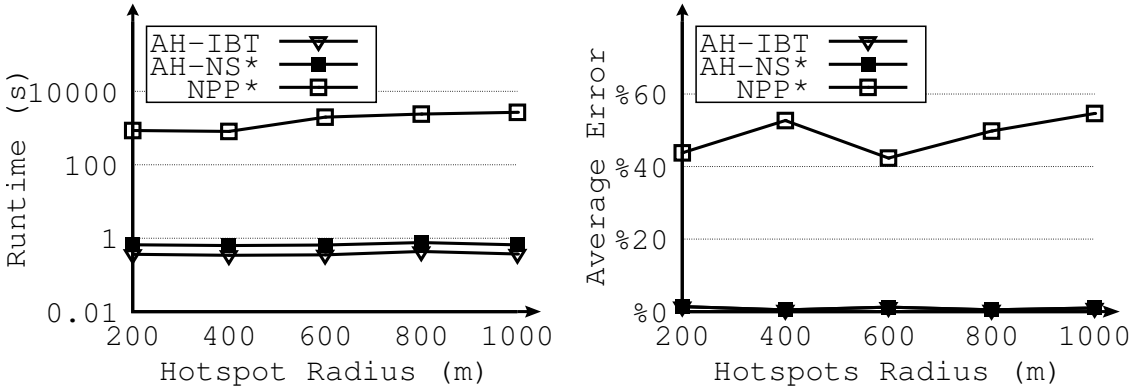
Figure .3a and Figure .4a show the runtime of AH-IBT, AH-NS*, and NPP* using Synth-Detroit-Hotspots dataset defined in Table 4.1. Both figures show that the number of hotspots and the radii of hotspots have a slight impact on the runtime for all algorithms. For similar reasons discussed in Section 4.6.3, AH-IBT consistently achieves better performance and it is up to 4 orders of magnitude faster than NPP* and up to 1.88 times faster than AH-NS*. Figure .3b and Figure .4b show the error of AH-IBT, AH-NS*, and NPP*. AH-

IBT and AH-NS* achieve better accuracy compared to NPP*, with up to 50%, for the same reason discussed in Section 4.6.3.



(a) Runtime (b) Accuracy

Figure .3: Effect of Varying Hotspot Numbers in HDWPR



(a) Runtime (b) Accuracy

Figure .4: Effect of Varying Hotspots Radii in HDWPR