Data Science for Mining Patterns in Spatial Events

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

Xun Tang

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

Advisor: Professor Shashi Shekhar

May, 2019

Acknowledgements

I show my deepest appreciation to my advisor, Prof. Shashi Shekhar, for the guidance and support throughout my Ph.D. The scientific research philosophy, vision and communication skills I have learnt from him are priceless to me.

I would like to thank Prof. George Karypis, Prof. Ravi Janardan, Prof. Snigdhansu Chatterjee, and Prof. Brent Hecht who served on my exam committees as well as my collaborator Dr. Christopher Farah for providing me detailed suggestions and feedbacks to shape my research.

I am grateful for the hardworking and smart members in the Spatial Computing Research Group: Xun, Dev, Kwangsoo, Viswanath, Zhe, Emre, Reem, Yiqun, Yan, Jayant, and Arun. All the meetings, discussions, and conversations with you made my Ph.D. more productive and less stressful.

I thank Kim Koffolt for reviewing my papers and improving my scientific writing skills.

Dedication

To my family for their unconditional support and sacrifice.

Abstract

There has been an explosive growth of spatial data over the last decades thanks to the popularity of location-based services (e.g., Google Maps), affordable devices (e.g., mobile phone with GPS receiver), and fast development of data transfer and storage technologies. This significant growth as well as the emergence of new technologies emphasize the need for automated discovery of spatial patterns which can facilitate applications such as mechanical engineering, transportation engineering, and public safety.

This thesis investigates novel and societally important patterns from various types of large scale spatial events such as spatial point events, spatio-temporal point events, and spatio-temporal linear events. Multiple novel approaches are proposed to address the statistical, computational, and mathematical challenges posed by different patterns. Specifically, a neighbor node filter and a shortest tree pruning algorithms are developed to discover linear hotspots on shortest paths, a bi-directional fragment-multi-graph traversal is proposed for discovering linear hotspots on all simple paths, and an apriori-graph traversal algorithm is proposed to detecting spatio-temporal intersection patterns.

Extensive theoretical and experimental analyses show that the proposed approaches not only achieve substantial computational efficiency but also guarantee mathematical properties such as solution correctness and completeness. Case studies using real-world datasets demonstrate that the proposed approaches identify valuable patterns that are not detected by current state-of-the-art methods.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Spatial data has explosively grown over the last decades thanks to the proliferation of location-based services (e.g., Google Maps) and corresponding devices (e.g., mobile phones with GPS receivers) as well as the development of fast data transfer and storage technologies [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. As a result, there is a great need for automated discovery of spatial patterns and corresponding data-driven approaches to benefit many societally important applications in mechanical engineering [27], transportation engineering [1], and public safety [28], etc. For example, using on-board diagnostic data (e.g., power, fuel consumption) collected from millions of vehicles on the road, previously unknown patterns that consider both environmental factors and engine behavior can be identified [27, 29, 30]. These patterns assist in designing new generation engines which substantially decrease fuel consumption and air pollution as well as novel eco-friendly navigation systems.

1.1 Spatial Data Mining

Spatial data mining studies the process of discovering interesting, previously unknown, but potentially useful patterns from large spatial databases [31, 32, 33, 34, 35, 12]. Figure 1.1 shows the entire knowledge discovery process. The core components are spatial data mining algorithms which take input spatial data and produce desired output pattern families, including spatial outliers, spatial interactions, spatial predictive models,

spatial partitions and summarizations, and hotspots. These algorithms have statistical foundations and integrate scalable computational techniques which follow certain mathematical properties such as correctness and completeness.

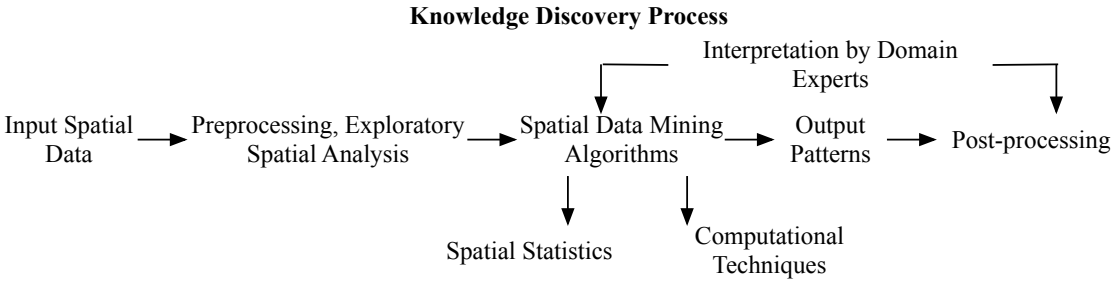


Figure 1.1: The process of spatial data mining

add: how is it different from traditional data mining (input data, auto-correlation, pattern families)

1.1.1 Societal Importance

Spatial data mining plays a critical role in many applications domains and across various governmental agencies. For example, NASA needs spatial data mining tools to classify earth observation images into land cover maps for ecology and environment management [36]. The National Institute of Health (NIH) uses spatial data mining tools to detect disease outbreaks for public health [37]. Law enforcement officers leverage spatial data mining techniques to help discover crime patterns [28]. The US Department of Transportation analyzes patterns (e.g., accident hotspots) in traffic data to improve transportation safety and efficiency [1]. Also, vehicle GPS trajectories together with engine measurements (e.g., fuel consumption, emission) are mined to recommend routes that are fast, fuel efficient, or environmental friendly [29, 30]. There are also other application domains such as earth science, climatology, precision agriculture, and the Internet of Things. The US Coast Guard is interested in mining trajectory patterns from Automatic Identification System (AIS) data [38] to help detect violations of maritime regulation such as illegal, unreported and unregulated (IUU) fishing and illicit

transactions [10].

The interdisciplinary nature of spatial data mining requires that its techniques be developed with an awareness of domain knowledge (e.g., underlying physics, theoretical assumptions) from the application disciplines [34, 39]. For example, climate science research finds that observable predictors for climate phenomena discovered by purely data driven methods can be misleading without consideration of climate models, locations, and seasons [40].

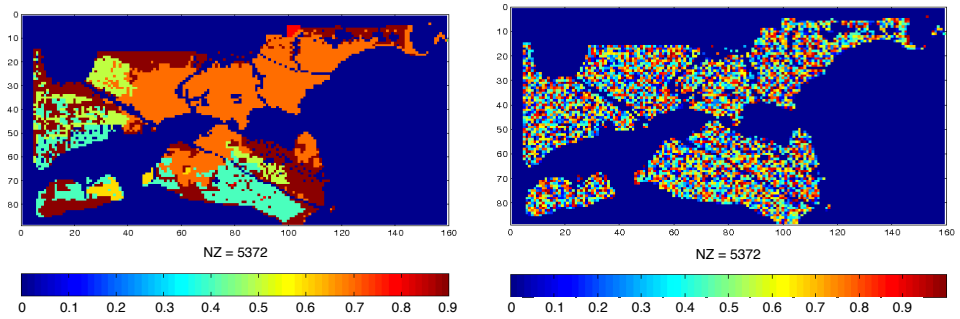
1.1.2 Spatial Statistical Foundations

Spatial statistics [41, 42, 43, 44] is a branch of statistics focusing on the analysis and modeling of spatial data. It is special due to the unique characteristics of space. For example, according to the first law of geography: "Everything is related to everything else, but near things are more related than distant things" [45]. Known as spatial autocorrelation, such spatial dependency across nearby locations violates the common assumption in classical statistics that variables are independent and identically distributed.

An illustration of the spatial autocorrelation phenomenon in the real world can be seen in Figure 1.2. Figure 1.2(a) shows a map of the distribution of vegetation durability across a marshland. The spatial autocorrelation effect can be seen from the fact that neighbor regions tend to have similar colors. By contrast, Figure 1.2(b) shows what the map would look like if vegetation durability followed an independent and identical distribution, which looks very different from the reality.

A spatial point process is a stochastic process for mining patterns from spatial point data. Unlike point reference data, the random variables are locations rather than non-spatial attributes. Examples include locations of crime events in a city. One basic type of spatial point process is a homogeneous spatial Poisson point process (also called complete spatial randomness, or CSR) [43], where point locations are independent following a homogeneous intensity over space. Figure 1.3 shows an example data set following CSR.

However, real world spatial point processes are often not completely spatial random. First, spatial aggregation (clustering) or spatial inhibition between point locations often exists as opposed to complete spatial independence as in CSR. Spatial statistics such as Ripley's K function [46] (the average number of points within a certain distance



(a) Actual vegetation durability distribution with spatial autocorrelation

(b) Vegetation durability distribution with i.i.d. assumption

Figure 1.2: A real world example of spatial autocorrelation effect

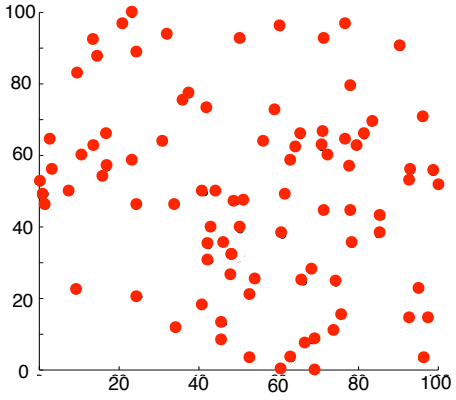


Figure 1.3: An example of complete spatial randomness

of a given point normalized by the overall intensity) can be used to test for spatial aggregation within a point pattern. Second, real world spatial point processes such as crime events often contain hotspot (significantly high intensity) areas instead of following homogeneous intensity across space. A spatial scan statistic [47] can be used to detect these hotspot patterns. The statistic uses a regular shaped window to scan the study area and test if point intensity within the window is significantly higher than outside. Though both the K-function and spatial scan statistics employ the same null hypothesis of CSR, their alternative hypotheses are quite different: The K-function tests whether points exhibit spatial aggregation or inhibition rather than independence, while spatial

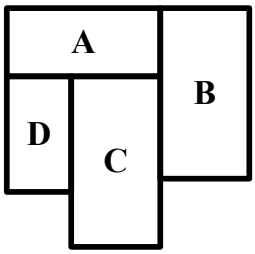
scan statistics assume that points are independent and test whether a hotspot with much higher intensity exists. Finally, there are other spatial point processes such as the Cox process, in which the intensity function itself is a random function over space, as well as a cluster process, which extends a basic point process with a small cluster centered on each original point [43].

Besides spatial point process which deals with spatial point data, other spatial statistic models have been proposed for other types of data as well [41, 42, 43, 44].

Geostatistics [44] deals with the analysis of the properties of point reference data, including spatial continuity (i.e., dependence across locations), weak stationarity (i.e., first and second moments do not vary with respect to locations) and isotropy (i.e., uniformity in all directions). For example, under the assumption of weak stationarity (or more specifically intrinsic stationarity), the variance of the difference of non-spatial attribute values at two point locations is a function of point location difference only, regardless of the specific locations of the points. This function is called a variogram [41]. If the variogram only depends on distance between two locations (not varying with respect to direction), it is further called isotropic. Under the assumptions of these properties, geostatistics also provides a set of statistical tools such as Kriging [41], which can be used to interpolate non-spatial attribute values at unsampled locations. In addition, real world spatial data may not always satisfy the stationarity assumption. For example, different jurisdictions tend to produce different laws (e.g., speed limit differences between Minnesota and Wisconsin). This effect is called spatial heterogeneity or non-stationarity. Special extended models (e.g., geographically weighted regression, or GWR [48]) can be used to model the varying coefficients at different locations. For example, in traditional linear regression model: $y = X\beta + \epsilon$, weight β and noise ϵ are location independent. However, in spatial linear regression, they are location dependent and vary over different locations.

Lattice statistics analyzes and models non-spatial attributes in a lattice (areal) model. One key issue in lattice statistics is to model the dependency across nearby cells. The range of such dependency is often represented by a neighborhood (or contiguity) matrix called a W-matrix. W-matrix elements (spatial neighborhood relationships) can be defined based on spatial adjacency (e.g., rook or queen neighborhoods), Euclidean distance, or in more general models, cliques and hypergraphs [49]. Figure 1.4(a) shows

an example of four cells and Figure 1.4(b) gives the W-matrix of these cells. If two cells are neighbors, the corresponding element in the matrix is 1. Figure 1.4(c) gives a normalized W-matrix in which the total values in each row is 1 unless all elements are 0 in the row.



	A	B	C	D
A	0	1	1	1
B	1	0	1	0
C	1	1	0	1
D	1	0	1	0

	A	B	C	D
A	0	0.33	0.33	0.33
B	0.5	0	0.5	0
C	0.33	0.33	0	0.33
D	0.5	0	0.5	0

(a) Input cells (b) W-matrix (c) Normalized W-matrix

Figure 1.4: An example of W-matrix

Based on a W-matrix, spatial autocorrelation statistics can be defined which reflect the correlation of non-spatial attribute values at neighboring locations. Common spatial autocorrelation statistics include Moran's I, Getis-Ord $Gi*$, Geary's C, Gamma index Γ [44], etc., as well as their local versions called local indicators of spatial association (LISA) [50]. A high spatial autocorrelation means non-spatial attribute values at nearby locations tend to be very similar. There are also several lattice statistics models, such as the spatial (or simultaneous) autoregressive model (SAR) and conditional autoregressive model (CAR), both of which belong to a more general family called Markov random fields (MRF). SAR directly models spatial dependency across response variables at different locations, while CAR models the dependency in conditional probability. A CAR model is often used as an additional term for capturing spatial autocorrelation in other Bayesian hierarchical models [41]. Another important issue in lattice statistics is the modifiable areal unit problem (MAUP) (also called the multi-scale effect) [51], which tells us that the same analysis method will show various results on different aggregation scales. For example, analysis using data aggregated by states will differ from analysis using data aggregated by counties.

1.1.3 Spatial Pattern Families

Spatial data mining outputs various patterns depending on the application needs and computational techniques. Now we go through some example pattern families that have been widely studied and applied, namely spatial outliers, co-locations, and spatial hotspots.

Spatial outliers: Global outliers are observations that appear to be inconsistent with remaining observations, or which deviate so much from other observations as to arouse suspicions that they were generated by a different mechanism. In contrast, a spatial outlier [52, 53] is a spatial object whose non-spatial attributes deviate significantly from those of its spatial neighbors. Informally, a spatial outlier is a local *instability* or *discontinuity*. For example, a new house in an old neighborhood of a growing metropolitan area is a spatial outlier but not a global outlier based on house age. In the county-level voting results of the 2012 U.S. presidential election in Florida shown in Figure 1.5, the lone county in the upper middle of the map voted for Obama while all its neighbors voted for Romney can be considered a spatial outlier.

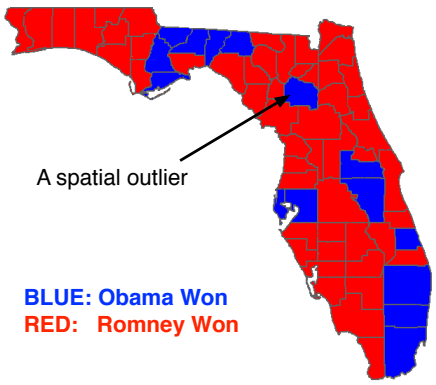


Figure 1.5: An illustrative example of a spatial outlier: voting results of 2012 U.S. president election in Florida

In transportation, spatial outlier indicates anomalous traffic patterns from sensor observations on a highway road network. In mobile banking, spatial outlier detection can help provide early warning for fraud transactions.

Spatial outlier detection approaches fall into two categories: visualization approaches

and neighborhood approaches. Visualization approaches identify spatial outliers via plotting of spatial locations on a graph. Common methods are variogram clouds [54] and Moran scatterplots [50, 44] as introduced earlier.

Neighborhood approaches begin by defining a spatial neighborhood, and then compute a test statistic (i.e., outlier score) based on the difference on non-spatial attributes between the current location and the neighborhood aggregate [52]. Spatial neighborhoods can be defined by geographical distances (e.g., K nearest neighbors), or by topological connectivity (e.g., adjacent nodes on road networks). The fundamental spatial outlier detection methods have been extended in a number of directions. Typical approaches focus on numerical attributes whose values are naturally ordered and thus able to be arithmetically operated (e.g., housing price). However, they are not able to handle categorical attributes such as housing colors. To solve the categorical spatial outlier problem, approaches based on measuring correlation between the occurrence pattern of categorical attribute values have been proposed [55, 56]. Spatial outlier detection with multiple non-spatial attributes [57, 58] finds a way to combines different non-spatial attributes to obtain the comprehensive outliers. Weighted spatial outlier detection [59] takes the impact of the neighbors of an object into consideration. Local variance-aware spatial outlier detection approach [60] considers the heterogeneity of variance, that is, the outlier score is computed based on not only the extent of variance but the average variance in that area.

Co-locations: Co-location patterns [61], represent subsets of spatial event types whose instances are often located in close geographic proximity. Real-world examples include symbiotic species in ecology, e.g., the Nile Crocodile and Egyptian Plover and commercial localization, e.g., fast food restaurants tend to open near other fast foot restaurants. Figure 1.6 shows an example of locations of Wendy's ("W"), KFC ("K"), and Pizza Hut ("P") in Manhattan, New York City. Some patterns that stores are open close to each other are observed, namely "W-K-P", "P-K", and "W-K".

Discovering various patterns of spatial co-location is important in applications related to ecology, environmental science, public safety, and climate science. For example, identifying spatially co-located crime types can help police departments to understand crime generators in a city, and thus take effective measures to reduce crime events.

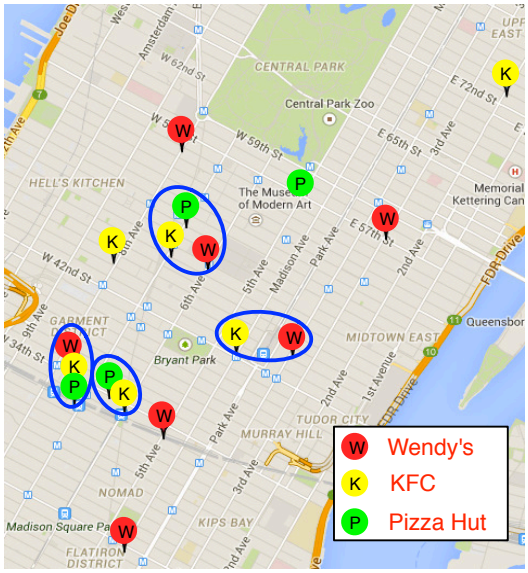
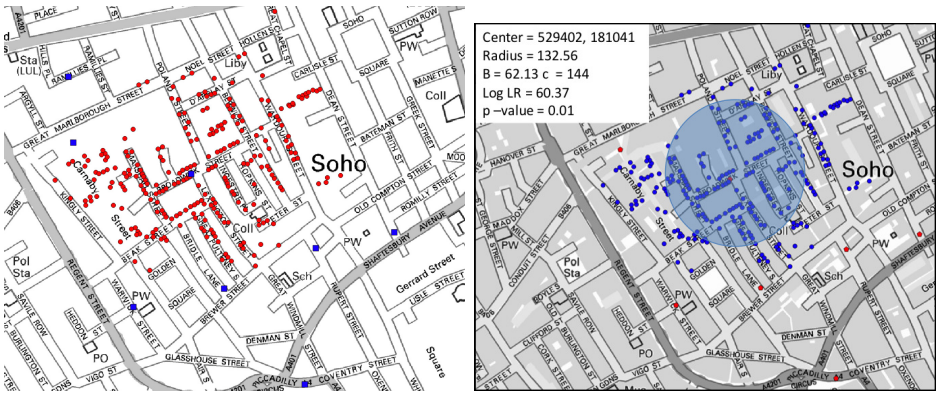


Figure 1.6: Illustration of Point Spatial Co-location Patterns. Different colors represent different spatial feature types. together

Different computational approaches have been developed to identify spatial co-locations. Traditional approaches use the cross-K function with Monte Carlo simulation [44], mean nearest-neighbor distance, and the spatial regression model [62]. Other approaches use an event centric model [61] based on a defined monotonic interest measure, participation index, which is an upper bound of cross-K function statistics, allowing efficient algorithms via apriori-based upper-bound pruning. Within the same framework, another interest measure, called the probabilistic participation index [63] deals with the uncertainty of spatial features caused by over-counting of the data. Also, maximal participation ratio [64] has been proposed to find patterns that involve object types having mismatched frequency. In order to take spatial heterogeneity into account, researchers have developed regional spatial co-location detection approaches which find object types that are highly correlated in certain sub-regions [65, 66, 67]. To minimize the cost of false positives, approaches that use statistical significance test to eliminate chance patterns have also been proposed [65, 68].

Spatial hotspots: Spatial hotspot detection finds regions where the number of events is significantly high. Figure 1.7 shows a real world spatial hotspot example from

the 1854 London Cholera outbreak detected by a popular tool, SaTScan [47, 69].



(a) Input Cholera cases (b) Circular hotspot detected by SatScan

Figure 1.7: A real world example of spatial hotspot in 1854 London Cholera outbreak

Spatial hotspot detection helps find disease outbreaks which allows officials to allocate resources to limit its spread [47, 69]. In criminology, finding crime hotspots may help police officials distribute their force on most critical regions and even help search for a criminal.

Spatial scan statistics [47, 69], the most popular spatial hotspot detection approach assumes that activities diffuse isotropically on Euclidean space and this it detects statistically significant hotspots in a circular/elliptical shape. A number of newer approaches [70, 71, 72, 73, 74] have been proposed which overcome the limitation of isotropic diffusion. More recent approaches deal with hotspots of events occurring on spatial networks [75, 76, 77]

Other approaches include Kernel density estimation (KDE) [78] which identifies hotspots via a density map of point events. First, a grid is created over the study area and a kernel function is used with a user-defined radius (bandwidth) on each point to estimate the density of points on centers of grid cells. A subset of grid cells with high density are returned as spatial hotspots. The Geographical Analysis Machine (GAM) method [79] first creates a regular grid in the study area, draws circles with a fixed radius around the grids, and identifies the 'dense' circles whose numbers of points are significantly higher than expected values. Finally, GAM merges overlapping dense circles and connected components as spatial hotspots.

1.2 Challenges

The challenges of spatial data mining come from three aspects: statistics, computing, and mathematics.

Spatial data mining poses unique statistical challenges due to the special characteristics of spatial data. First, spatial data are embedded in continuous space and time, and thus many classical data mining techniques assuming disjoint data (e.g., transactions in association rule mining) may not be effective. Second, spatial data exhibits spatial autocorrelation effects. Ignoring autocorrelation and assuming an identical and independent distribution (i.i.d.) when analyzing spatial data may produce hypotheses or models that are inaccurate (e.g., salt and pepper noise, pixels predicted as different from neighbors by mistake) [80]. Third, interactions are non-isotropic (varying across spatial directions) and may exist in multiple spatial scales. In addition to spatial dependence at nearby locations, phenomena of spatial tele-coupling also indicate long range spatial dependence such as El Nino and La Nina effects in the climate system. Finally, spatial data exhibits spatial heterogeneity and temporal non-stationarity, i.e., identical feature values may correspond to distinct class labels in different spatial regions.

The computational challenges are due to the huge amount of spatial data and the cost of identifying complex patterns. For example, on-board diagnostic data collected every day from millions of running vehicles can be in petabytes and contain hundreds of variables (e.g., speed, emission). Identifying the combinations of negatively correlated variables requires enumerating exponential of the number of variables, leading to intractable computational cost. In addition, statistical significance test needs to be enforced in certain patterns due to the potentially unacceptable cost from false positives. For example, a falsely identified crime hotspot may cause panic and a drop in house prices in the neighborhood.

Spatial data mining algorithms need to ensure necessary mathematical properties of the output patterns such as correctness and completeness. Correctness indicates that the algorithm is able to terminate and the output patterns must all the solution requirements. For example, a hotspot must pass the statistical significance test (i.e., p-value $\leq$ threshold) to be considered an output. Completeness indicates that no patter that meets all the solution requirements is missed in the output. For example, the

spatial scan statistics [47, 69] detects circular hotspots defined by two events, one at the center and the other on the circumference. To ensure completeness, it needs to enumerate all such circles defined by two points. Completeness is always associated with a specific pattern definition which typically involves a trade-off between computational tractability and richness of the pattern. For example, even though the spatial scan statistics [47, 69] achieves completeness for circular hotspots defined by two points, it is not able find any hotspots whose center has no event. Worse, it misses all the hotspots that are not circular.

1.3 Thesis Contributions

This thesis investigates data science techniques for mining patterns from various footprints of spatial events. The contributions are summarized in a taxonomy shown in Table 1.1 which describes the input spatial events and output spatial patterns classified by their footprints. The input spatial events are classified into four types: First, a spatial point event is modeled as a static 2-D point (x, y) representing the location where the event occurs and the temporal information is omitted. For example, all traffic accidents occurring on a street over a year can be modeled as a collection of spatial point events. One step beyond the spatial point event is the a spatio-temporal point, which takes temporal information into account and can be modeled as a 3-D point (x, y, t) representing an event located at (x, y) at time t. For example, when studying the pattern of traffic accidents on a street at different times of day, a traffic accident can be modeled as a spatio-temporal point event. The third event footprint studied is the spatio-temporal linear event, which is represented a sequence of N 3-D points $(x_i, y_i, t_i), i = 1...N$ which are typically extracted from trajectories. For example, the trajectory of a vehicle speeding during a period of time can be modeled as a spatio-temporal linear event. Lastly, we examined the spatio-temporal event in areal data, which occurs on a sequence of matrices of pixels where each pixel represents attributes in the corresponding space and time. One example is a spreading forest fire that appears on remote sensing imagery. As with the input events, the output spatial patterns are classified as point, linear, or areal patterns. Based on the different complexities of the enumeration space, linear patterns can be further classified into shortest paths, simple paths, or all paths.

This thesis proposes several novel spatial data mining approaches (i.e. linear hotspot discovery on shortest paths, linear hotspot discovery on all simple paths, and spatio-temporal intersection detection from trajectories with data gaps) which address the statistical, computational, and mathematical challenges posed by various footprints of input spatial events and output patterns. Not only do the proposed approaches achieve high computational efficiency but also the statistical and mathematical properties are well studied. The content of each chapter is briefly introduced below.

Table 1.1: A taxonomy of thesis contributions

ST: Spatio-temporal		Pattern footprint				
		Point	Linear			Areal
			Shortest paths	Simple paths	All paths	
Event footprint	Spatial point event		Linear hotspots on shortest paths (**Chapter 2**)	Linear hotspots on all simple paths (**Chapter 3**)		
	ST point event					
	ST linear event in trajectory					ST Intersections (**Chapter 4**)
	ST event in areal data					

- Chapter 2 presents a novel hotspot detection approach, Linear Hotspot Discovery on Shortest Paths (LHDSP), which finds all shortest paths on a spatial network where the concentration of input spatial point events is statistically significantly high. The method addresses the limitation of traditional spatial hotspot detection approaches only focusing on hotspots in Euclidean space. LHDSP has many societal application fields such as transportation planning where the events (e.g., traffic accidents) are usually associated with linearly shaped roadways. LHDSP proposes multiple novel algorithms, namely neighbor node filter and shortest path

tree pruning, to successfully handle the computational challenges from large number of candidate paths, non-monotonicity of the test statistic, and statistical significant test. To balance between computational tractability and richness of the pattern, LHDSP assumes that most travelers prefer to move along shortest paths. Under this assumption, LHDSP enumerates all shortest paths on a given spatial network and guarantees correctness and completeness of the solution. Theoretical and experimental analyses show that the proposed algorithms yield substantial computational savings compared to baseline approaches. Case studies demonstrate that LHDSP can find novel linear patterns that have never been detected previously.

- Chapter 3 investigates the problem of Linear Hotspot Discovery on All Simple Paths (LHDA) which considers much richer patterns compared to LHDSP (proposed in Chapter 2) does by enumerating all simple paths on a given spatial network. To address this much more difficult computational challenge (i.e. #-p hard), LHDA proposes a novel algorithm based on bi-directional fragment-multigraph traversal (i.e. ASP_FMGT). Extensive theoretical and experimental analyses show that ASP_FMGT has substantially improved performance over a baseline approach using depth-first-search with backtracking while keeping the solution correct and complete. Case studies on real-world datasets show that ASP_FMGT outperforms existing approaches including LHDSP since it discovers new hotspots unknown before and achieves higher accuracy for locating known hotspots.

- Chapter 4 investigates the problem of Spatio-temporal Intersection Detection From Trajectories With Data Gaps (STID) which aims to identify meet-up patterns between objects (e.g., ships, vehicles) whose trajectories have missing segments (i.e. Spatio-temporal Intersection (STI) pattern) from large scale spatio-temporal linear events. The applications of STID include maritime safety and regulation, homeland security, and public safety. STID overcomes the limitation of traditional trajectory mining approaches which assume the availability of trajectory data or interpolate the missing segments. It also addresses the computational challenge from the huge amount of trajectory data and the high complexity of the STI pattern with an apriori-graph traversal based algorithm AGT_STID.

Both theoretical and experimental analyses show that AGT_STID significantly increases the computational performance over baseline approaches and guarantees solution correctness and completeness. A case study on real-world maritime trajectory data demonstrates AGT_STID is able to discover previously unknown meet-up patterns and thus reveal possible illicit maritime transaction activities.

- Chapter 5 summarizes the thesis findings and gives an overview of related directions and topics for research in the future.

Chapter 2

Linear Hotspot Discovery on Shortest Paths

2.1 Introduction

Linear Hotspot Discovery on Shortest Paths (LHDSP) identifies routes with statistically significant concentrations of activities (e.g., crimes, accidents, etc.). Informally, the LHDSP problem can be defined as follows: given a spatial network, a collection of geo-referenced activities (e.g., pedestrian fatality reports, crime reports), and a concentration of activities threshold θ_d, find all shortest paths between activities in the spatial network where the concentration of activities is unusually high (i.e., statistically significant) and the concentration of activities is equal to or greater than θ_d. Depending on the domain, an activity may be the location of a pedestrian fatality, a carjacking, a train accident, etc. Figure 2.1 illustrates an input example of LHDSP consisting of 7 nodes, 7 edges (with edge weights set to 1 for illustration purposes), 10 activities (shown as squares on the edges), and $\theta_d = 12$, indicating that we are interested in the shortest paths between activities whose concentrations of activities is equal to or greater than $\theta_d = 12$.

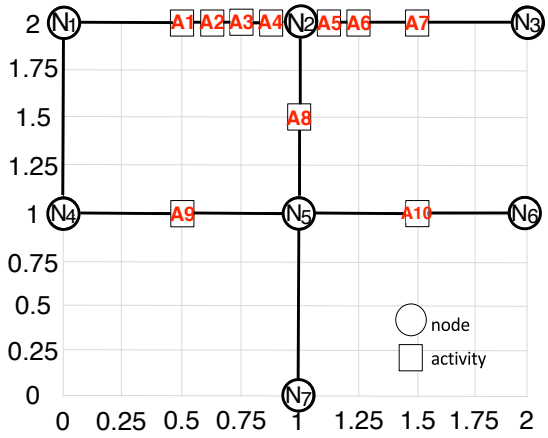


Figure 2.1: Example of input of Significant Linear Hotspot Discovery (Best in color).

2.1.1 An Illustrative Application Domain: Preventing Pedestrian Fatalities

Urban computing aims to handle the major issues that cities face by using computational techniques. Seven major urban computing categories are urban planning, transportation, environment, energy, social, economy, and public safety and security [81]. Our proposed work focuses on discovering the statistically significant linear hotspots on road networks to address the issues associated with two of these application domains namely transportation planning and public safety and security.

To illustrate the applicability of Linear Hotspot Discovery on Shortest Paths (LHDSP), we focus on the problem of discovering significant concentrations of pedestrian fatalities in a transportation network. According to a recent policy report, more than 47,700 pedestrians were killed in the United States from 2000 to 2009 [1], and more than 688,000 pedestrians were injured over the same time period, which is equivalent to a pedestrian being struck by a vehicle every 7 minutes. Pedestrian fatalities have increased in many places, including 15 of the country's largest metro areas, even as overall traffic deaths have fallen [1].

Domain experts attribute pedestrian fatalities largely to the design of streets, which have been engineered for speeding traffic with little or no provision for people on foot, in wheelchairs or on bicycles [1]. Daily activities have shifted away from city streets towards

(a) (b) (c)

Figure 2.2: (a) Pedestrian at risk on a road without proper sidewalks [1]. (b) Paved sidewalk and road separators for pedestrians, bicycles, and motorcycles build on road [2]. (c) Pedestrian fatalities occurring on arterials in Orange County, FL [3] (Best in color).

higher speed arterials. This has resulted in more than half of fatal pedestrian crashes occurring on these wide, high capacity and high-speed thoroughfares. Typically designed with four or more lanes and high travel speeds, arterials are not built with pedestrians in mind (Figure 2.2(a)). They lack sidewalks, crosswalks (or have crosswalks spaced too far apart), pedestrian refuges, street lighting, and school and public bus shelters [1].

This lack of basic infrastructure can be lethal. For example, forty percent of fatalities occurred where no crosswalk was available [1]. Figure 2.2(c) shows a map of pedestrian fatalities that occurred on Orange County roads from 2000 to 2009. Transportation planners and engineers need tools to assist them in identifying which frequently used road segments/stretches pose statistically significant levels of risk for pedestrians and consequently should be redesigned. A promising way to effectively discover such road segments/stretches raises from the availability of the large amount of fatality data and corresponding data analysis approaches [81]. Road segments/stretches that pose significant risks to pedestrians may be conceptualized as linear concentrations because the generation model of pedestrian fatalities is inherently linear, i.e., they occur on roads. This chapter presents an approach for identifying statistically significant linear concentrations of activities such as pedestrian fatalities in a spatial network.

If traditional hotspot discovery is used (e.g., circular hotspot detection), the government can just fix the hotspots for the pedestrian. However, often such hotspots do not occur on a circular area and the fix of those hotspot locations requires a road to be analyzed for structural mistakes by the transportation planners. The Minnesota

Department of Transportation proposes that building lane separators for pedestrian sidewalks, bicycles and motorcycles along roads illustrated in Figure 2.2(b) as a way to decrease the risk of traffic accidents [2]. In addition, lack of barriers between opposite traffic on specific parts of highways may lead to severe traffic accidents. For example, since Spring 2015, there were two fatal traffic accidents occurred on Minnesota Highway 280 at similar locations. The cars that lost control went to the flowing traffic on the opposite side since no barriers were in between along the highway segment [82].

Traditional network-based techniques which consider edges atomic face a fundamental limitation in dealing with very long edge with a dense cluster of activities at one end of the edge. These techniques may either fail to capture the precise locations of the hotspots on such edges or even completely miss the hotspots. For example in Figure 2.1, $\langle N_1, N_2 \rangle$ is an edge whose activities are densely clustered near N_2 (i.e., A_1, A_2, A_3, A_4). Traditional techniques may capture $\langle N_1, N_2 \rangle$ as a hotspot, however, the precise location which is actually $\langle A_1, A_4 \rangle$ is not captured. In worse case, suppose the concentration of $\langle A_1, A_4 \rangle$ exceeds the threshold but the overall concentration of the entire path $\langle N_1, N_2 \rangle$ does not since it is compromised by the empty end $\langle N_1, A_1 \rangle$, this hotspot will be completely missed. To address this limitation, domain experts introduces *dynamic segmentation* which segments edges into sub-edges at each activity. This chapter investigates *dynamic segmentation* in order to evaluate if paths between activities are hotspots. For example, $\langle A_1, A_4 \rangle$ will be returned as a hotspot with a more precise location than $\langle N_1, N_2 \rangle$ returned by the traditional techniques. In the other case that $\langle N_1, N_2 \rangle$ is not qualified as a hotspot, $\langle A_1, A_4 \rangle$ may also be return as a hotspot. The technical details of *dynamic segmentation* are elaborated in the problem statement (Section 2.2). The significance of using *dynamic segmentation* is demonstrated in the case studies on real datasets (Section 5).

Linear hotspot discovery can also be applied to other application domains. Certain types of crimes (e.g., assaults, street robbery) may form hotspots on transportation networks that represent roads needing more attention from police departments [3]. In addition, water quality changes on river networks may form hotspots that represent bursts of pollution [83].

2.1.2 Challenges

LHDSP is challenging due to the potentially large number of candidate routes ($\sim 10^{16}$) in a given dataset with millions of activities and road segments. Enumerating and evaluating all shortest paths at the sub-edge level results in prohibitive computational cost. Additionally, density ratio does not obey the monotonicity property, meaning that there is no ordering between the density ratio of a path and its super-paths, or vice-versa. Furthermore, depending on the method used to determine statistical significance, computation times may also be impacted (e.g., $m = 1000$ Monte Carlo simulations may be required to calculate statistical significance).

2.1.3 Related Work and Their Limitations

Dividing spatial data into statistically significant groups is an important task in many domains (e.g., transportation planning, public health, epidemiology, climate science, etc.). Methods for this type of partitioning may generally be considered to be geometry-based or network-based.

Geometry-based techniques [84, 4] partition spatial data using geometric shapes (e.g., circles, rectangles). This is useful in domains such as public health, where finding spatial clusters with a higher density of disease is of interest for understanding the distribution and spread of diseases, outbreak detection, etc. Kulldorff, et al. proposed a spatial scan statistics framework (and the SaTScan software) for disease outbreak detection [47]. The spatial scan statistic employs a likelihood ratio test where the null hypothesis is the probability that disease occurrence inside a region is the same as outside, and the alternative hypothesis is that there is a higher probability of disease inside than outside. All the spatial regions, represented by a circle or ellipsoid in the spatial framework, are enumerated and the one that maximizes the likelihood ratio is identified as a candidate. However, if we apply SaTScan to a road network, many significant routes may be missed since a large fraction of the area bounded by circles for activities on a path will be empty. Figure 2.3(a) shows an example of the output of SatScan. Two circular hotspots are detected with large empty areas, which result in high p-values (i.e., 0.125 and 0.497). Furthermore, geometry-based techniques may not be appropriate for modeling linear clusters, which are formed when the underlying

generator of the phenomena is inherently linear (e.g., pedestrian fatalities, railroad accidents, etc.).

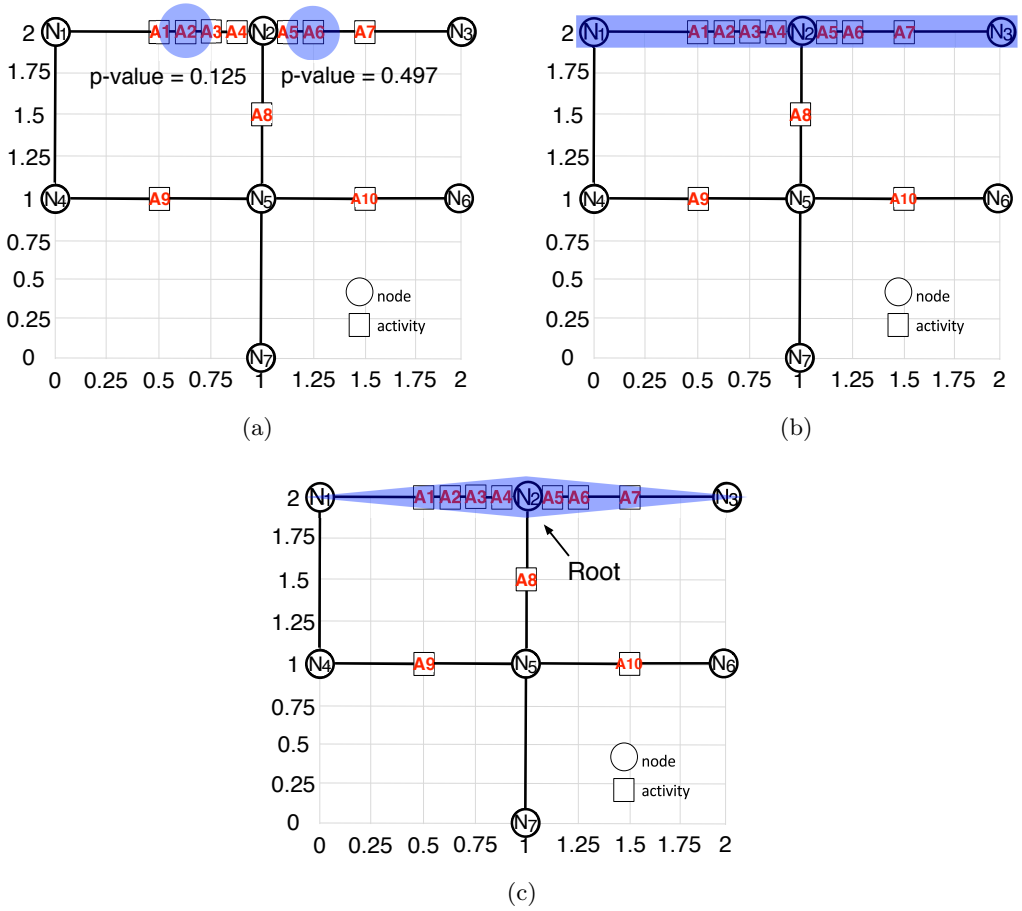


Figure 2.3: Example (a) Output of SatScan [4], (b) Output of Linear Intersecting Paths (LIP) [5], and (c) Output of Constrained Minimum Spanning Trees (CMST) [6] (Best in color).

By contrast, network-based techniques [6, 5, 85] leverage the underlying spatial network when partitioning spatial data. For example, Linear Intersecting Paths (LIP) [5] and Constrained Minimum Spanning Tree (CMST) [6] utilize a subgraph (e.g., a path or tree) to discover statistically significant groups.

LIP [5] discovers one anomalous sub-component of a set of connected paths that intersect each other. The connected paths are based on locations in the spatial network

with the highest percentage of activities, specified by the user. Hence the density ratio is evaluated only on a portion of the graph specified by this percentage, not on the entire spatial network. Figure 2.3(b)(b) shows an example of the output of LIP. The user-specified percentage is 40%, which means all the candidates will have paths containing edge $\langle N_1, N_2 \rangle$ since this edge has 4 activities (out of 10 activities). Examples of possible candidates are $\langle N_1, N_3 \rangle$, $\langle N_1, N_5 \rangle$, $\langle N_2, N_4 \rangle$, $\langle N_1, N_7 \rangle$, etc. The output is $\langle N_1, N_3 \rangle$, since it has the highest density ratio. However, in addition to returning only one statistically significant component, the results of this approach are sensitive to the percentage of the network selected. If the percentage is too high, the number of candidates may be highly restricted, which could result in not identifying statistically significant regions of interest. If the percentage is too low, LIP may be computationally prohibitive due to the large number of candidates.

Another network-based technique, CMST [6], finds one statistically significant tree in the spatial network. Figure 2.3(c) shows an example of the output of CMST. Here the output is $\langle N_1, N_3 \rangle$, where N_2 is the root, since this tree has the highest density ratio. However, this approach also has limitations. In addition to returning only one statistically significant tree, the size of the tree is restricted, which could result in not identifying statistically significant regions of interest.

2.1.4 Contributions

In this chapter, we present a new dynamic segmentation model which to the best of our knowledge is the first model that allows for discovering of multiple statistically significant routes at the sub-edge level in a spatial network. We present new algorithmic refinements (i.e., neighbor node filter, shortest path tree pruning) for sub-edge level linear hotspot discovery in a scalable way. We also present a cost model for the proposed algorithms and prove that our proposed algorithmic refinements are correct. Specifically, our research contributions are as follows:

- We propose a new model named dynamic segmentation, which allows the proposed approach to find multiple significant linear hotspots at the sub-edge level in the spatial network. We also introduce new algorithmic refinements to improve the scalability of linear hotspot detection with dynamic segmentation, including a

neighbor node filter and a shortest path tree pruning algorithm.

- We analytically prove the correctness of the proposed algorithms and present a cost analysis.
- We present two case studies comparing the detection results under dynamic segmentation with results in the related works, including SRM_GIS [86] and SatScan [4].
- Experimental results on real and synthetic data show that the proposed algorithms, yield substantial computational savings over SRM_GIS [86] without reducing either completeness or correctness of the result.

2.1.5 Scope and Outline of the Chapter

This chapter focuses on finding significant discrete activity events (e.g., pedestrian fatalities, crime incidents) associated with a point on a network. This does not imply that all activities must necessarily be associated with a point in a street. In addition, other network properties such as GPS trajectories and traffic densities of road networks [87, 88] are not considered. In this work, it is assumed that the number of activities on the road network is fixed and does not change over time. We do not consider techniques that do not employ statistical significance (e.g., DBScan [89], K-Means [90], KMR [91], and Maximum Subgraph Finding [92]). This chapter only enumerates shortest paths rather than all possible paths. This increases the computational tractability since the enumeration space decreases from exponential to polynomial. We choose shortest paths based on the assumption people generally prefer taking shortest paths a destination. Discovering hotspots on shortest paths may find the most possible "dangerous" routes between two locations.

The chapter is organized as follows: Section 2.2 presents the basic concepts and problem statement of Linear Hotspot Discovery on Shortest Paths (LHDSP). Section 2.3 presents our preliminary results towards addressing LHDSP. Section 2.4 details our proposed approaches for discovering sub-edge level linear hotspots in a scalable way. Section 2.5 presents two case studies comparing the proposed significant network-based outputs (i.e., shortest paths) to a significant geometry-based output (e.g., circles) on pedestrian fatality data. The experimental evaluation is covered in Section 2.6. Section 2.7 presents a discussion. Section 2.8 concludes the chapter and previews future

work.

2.2 Basic Concepts and Problem Statement

This section reviews several key concepts in LHDSP and presents a formal problem statement.

2.2.1 Basic Concepts

The basic concepts are defined as follows:

Definition 1 *A **spatial network** $G = (N, E)$ consists of a node set N and an edge set E, where each element n in N is associated with a pair of real numbers (x, y) representing the spatial location of the node in Euclidean space [93]. Edge set E is a subset of the cross product $N \times N$. Each element $e = (n_i, n_j)$ in E is an edge that joins node n_i to node n_j.*

Figure 2.1 shows an example of a spatial network where circles represent nodes and lines represent edges. A road network is an example of a spatial network where nodes represent street intersections and edges represent streets.

Definition 2 *An **activity set** A is a collection of activities. An **activity** $a \in A$ is an object of interest associated with only one edge $e \in E$ and has a location in Euclidean space.*

In transportation planning, an activity may be the location of a pedestrian fatality; in crime analysis, an activity may be the location of a theft. Some of the edges in Figure 2.1 are associated with a number of activities (e.g., edge $\langle N_1, N_2 \rangle$ has 4 activities).

Definition 3 *The **activity coverage inside a path**, a_p, is the number of activities on p. The **activity coverage outside** p is $|A| - a_p$, where $|A|$ is the total number of activities in the spatial network, G.*

For example, in Figure 2.1, the activity coverage *inside* path $\langle A_9, N_5, A_8, N_2, A_5, A_6 \rangle$ is 4 whereas the activity coverage *outside* this path is $10 - 4 = 6$.

Definition 4 *The **weight inside a path**, w_p, is the sum of weights of all edges in p. The **weight outside** p is $|W| - w_p$, where $|W|$ is sum of weights of all edges in G. In transportation planning, weight may represent distance or travel time of the path.*

In Figure 2.1, the weight *inside* $\langle A_9, N_5, A_8, N_2, A_5, A_6\rangle$ is 1.75 whereas the weight *outside* this path is $7 - 1.75 = 5.25$.

Definition 5 *The **density ratio of path** p, $\lambda_p = \frac{a_p \div w_p}{(|A|-a_p)\div(|W|-w_p)}$ [85, 47].*

The density ratio of path p, λ_p is the ratio of the activity density *inside* path p, $\frac{a_p}{w_p}$ to the activity density *outside* p, $\frac{|A|-a_p}{|W|-w_p}$. Table 2.1 lists 3 shortest paths from Figure 2.1, namely $\langle A_9, A_6\rangle$, $\langle A_4, A_{10}\rangle$, and $\langle A_1, A_7\rangle$. Path $\langle A_9, A_6\rangle$ contains activities A_9, A_8, A_5 and A_6 and has a weight of 1.75, hence its density is $\frac{4}{1.75}$ while the density outside is $\frac{10-4}{7-1.75}$. Therefore, the density ratio of path $\langle A_9, A_6\rangle$ is $\frac{4\div 1.75}{(10-4)\div(7-1.75)} = 2$. By similar calculation, path $\langle A_4, A_{10}\rangle$ has a density ratio of 1.42 and path $\langle A_1, A_7\rangle$ has a density ratio of 14.

The reason why we use density ratio as the test statistic in this chapter is two-fold. First, density ratio is in a family of metrics inspired from the hypothesis test in which the null hypothesis is "the density inside and outside a path are equal" while the alternative hypothesis is "the density inside a path is larger than outside". These metrics are largely used in hotspot detection literature [47, 84, 94] and follow three properties [84]: (1) Given a fixed weight, the metric increases monotonically with activity coverage. (2) Given a fixed activity coverage, the metric decreases monotonically with weight. (3) Given a fixed ratio of activity coverage to the weight, the metric increases monotonically with the weight. Any metrics that follow these properties (e.g., log likelihood ratio [47]) can be directly applied in the proposed approaches without any algorithmic changes. Second, among these metrics, density ratio is widely used in the literature that deals with activities associated with spatial networks [85]. We use density ratio to make it easier to compare the proposed algorithms with the literature.

Table 2.1: Examples of density ratio.

Path	Activities inside	Weight	Activities outside	Weight outside	Density ratio
<A_9,N_5,A_8,N_2,A_5,A_6>	4	1.75	6	5.25	2
<A_4,N_2,A_8,N_5,A_{10}>	3	1.625	7	5.375	1.42
<A_1,A_2,A_3,A_4,N_2,A_5,A_6,A_7>	7	1	3	6	14

Definition 6 *An **active edge** is an edge $e \in E$ that has 1 or more activities. An **active node** is a node u joined by an active edge. An **inactive node** is a node that is not joined by any active edges.*

Definition 7 *An **Active node** is a node $n \in N$ that at least one of its incident edges has 1 or more activities.*

Edges $\langle N_1, N_2 \rangle$ and $\langle N_2, N_3 \rangle$ in Figure 2.1 are active edges because they each have at least one activity, and nodes N_1, N_2, N_3, N_5, N_6, and N_7 are all active nodes because they are all joined by active edges. By contrast, Node N_4 is an inactive node because it is not joined by any active edges.

Definition 8 *A **super-path** of path p is any path sp that contains p, where sp is a subset of G. A **sub-path** is a path making up part of the super-path.*

For example, in Figure 2.1, $\langle N_1, N_2, N_5, N_6 \rangle$ and $\langle N_1, N_2, N_5, N_7 \rangle$ are super-paths of $\langle N_1, N_2, N_5 \rangle$. Conversely, $\langle N_1, N_2, N_5 \rangle$ is a sub-path of $\langle N_1, N_2, N_5, N_6 \rangle$.

2.2.2 Problem Statement

The problem of Linear Hotspot Discovery on Shortest Paths (LHDSP) can be expressed as follows:

Given:

1. A spatial network $G = (N, E)$ with a set of geo-referenced activities A, each if which is associated with an edge.

2. A density ratio threshold, θ_λ,

3. A p-value threshold, θ_p and the corresponding number of Monte Carlo simulations needed, m,

Find: All shortest paths $r \in R$ with $\lambda_r \geq \theta_\lambda$, p-value $\leq \theta_p$
Objective: Computational efficiency
Constraints:

1. $r_i \in R$ is not a sub-path of $r_j \in R$ for $\forall r_i, r_j \in R$ where $r_i \neq r_j$,
2. $\forall r_i \in R$ is not shorter than a minimum distance (ϕ) threshold θ_ϕ
3. Correctness and completeness

The spatial network input for LHDSP is defined in Definition 1. The θ_λ input is a threshold indicating the minimum desired density ratio. The θ_p input is the desired level of statistical significance and m is the corresponding number of Monte Carlo simulations needed for determining statistical significance. The output for LHDSP is all shortest paths between activities meeting the desired likelihood ratio and level of statistical significance. The shortest paths returned are constrained so that they are not sub-paths of any other path in the output. This constraint aims to improve solution quality by reducing redundancy in the paths returned. In addition, the distance of significant paths cannot be shorter than θ_ϕ. This constraint aims to avoid meaningless tiny paths that have high density ratio (e.g., a path between two activities very close to each other may have a high density ratio).

Dynamic Segmentation: Our approach resolves statistically significant routes to the sub-edge level (i.e., routes between activities), which is not investigated in our previous work [86]. This requires a model called dynamic segmentation. Intuitively, it modifies the traditional network structure such that new nodes are formed at the locations of activities and new edges are added to connect these nodes.

The pseudocode in Algorithm 1 shows the process of dynamic segmentation. All the edges with a activities (where $a > 0$) are split into a nodes and $a + 1$ edges (line 2). For clarification, in this model, the nodes before segmentation are referred to as static nodes, while the newly formed nodes, which are essentially activities, are referred to as dynamic nodes. The weights of the newly formed edges in the dynamically segmented network are then updated based on the distance between activities (line 3). In other

words, the weight of the dynamic edge formed between activity x and activity y will be updated to the distance between these two activities. Dynamic nodes are stored in *dynamicNodes*, and newly formed edges are stored in *dynamicEdges* (line 4).

For example, in Figure 2.1, activities A_1, A_2, A_3, etc. become nodes in the spatial network, and edge $\langle N_1, N_2$ is cut into several edges, namely $\langle N_1, A_1\rangle$, $\langle A_1, A_2\rangle$, $\langle A_2, A_3\rangle$, $\langle A_3, A_4\rangle$, and $\langle A_4, N_2\rangle$.

The dynamic segmentation model enables us to evaluate paths that start and end with activities and may be in the middle of an edge. As such, the density ratios tend to be more precise since the extra portions of the path before the first activity and after the last activity are trimmed. Therefore, segments which were previously not tested for statistical significance or which may have been previously deemed "not significant" because they were on a long empty edge, may end up as part of the result. For example, in Figure 2.1, $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ and $\langle N_1, N_2, N_3\rangle$ have the same set of activities but the weight of $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ is less. In this case, the density ratio for $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ is much higher (14 vs. 5.83), and the p-value is smaller (0.001 vs. 0.006).

Algorithm 1 Dynamic Segmentation

Input:

1) A spatial network $G = (N, E)$ with a set of geo-referenced activities with point locations on network nodes or edges and weight function $w(u, v) > 0$ for each edge $e = (u, v) \in E$ (e.g., network distance)

Output: A dynamically segmented spatial network G_s derived from G

Algorithm:

1: **for each** edges $e \in G$ with $a > 0$ activities **do**
2: Split e into a nodes, n_a, and $a + 1$ edges, e_a
3: update weights of e_a based on coordinates of activities
4: $dynamicNodes \leftarrow n_a$, $dynamicEdges \leftarrow e_a$

Finding Significant Paths

Each shortest path in the spatial network is evaluated for statistical significance using Monte Carlo simulations to determine whether or not it is statistically anomalous. Here the null hypothesis states that the paths identified by the path density ratio are random or by chance alone. The density ratio is associated with a p-value to decide whether the null hypothesis should be rejected in the hypothesis test. The p-value is the probability

of obtaining a density ratio that is equal to or greater than than that observed by chance alone.

In the Monte Carlo simulations, each activity in the original graph G is randomly put on a location in G so that the number of activities on each edge is shuffled, forming a new graph G_s. Note that all the activities in G are present in G_s, with no activities added or removed. We then compare the highest density ratio λ_{maxG_s} of randomized G_s with the density ratio of each path p_i whose density ratio is equal to or greater than the threshold in the original G. In a naïve way, in order to compute λ_{maxG_s} of G_s, all-pairs shortest paths in G_s need to be computed using Dijkstra's algorithm since shuffled activities are considered as nodes, making G_s a new graph. Then the density ratios of these paths are evaluated. However, an algorithmic refinement named neighbor node filter (Section 4.1.1) is proposed to evaluate the density ratios without running Dijkstra's algorithm on the whole graph. If the original value is smaller, then $c = c + 1$ for path p_i. The above process repeats m, making the subsequent p-value $\frac{c+1}{m}$ for path p_i. Paths whose p-values are less than or equal to the given p-value threshold are deemed statistically significant.

2.3 Preliminary Results

We initially solved the LHDSP problem with a previously proposed algorithm (SRM_GIS) [86] featuring two algorithmic refinements: Density Ratio Pruning and Monte Carlo Speedup. Note that even though SRM_GIS was proposed to solve LHDSP without dynamic segmentation, it can also be directly applied to dynamically segmented networks. Before describing SRM_GIS, we first review a naïve algorithm (SRM_Naïve) that solves the LHDSP problem.

2.3.1 Naïve Significant Route Miner (SRM_Naïve)

Algorithm 2 presents the pseudocode for the SRM_Naïve approach. The basic idea behind the algorithm is to find all statistically significant shortest paths in the dynamically segmented spatial network whose density ratio exceeds the threshold θ_λ, under the constraint that the shortest paths returned are not sub-paths of any other path in the output. Algorithm 2 proceeds by calculating all-pairs shortest paths, P, in the spatial

network (Line 2). Line 3 evaluates each shortest path in P to determine if it meets the given θ_λ to form a *Candidates* set. In line 4, the statistical significance of each shortest path in *Candidates* is evaluated and the significant routes are stored in *SigRoutes*. In order to assess statistical significance, all shortest paths in each of the m simulated graphs are used to calculate the p-value. In line 5, all paths in *SigRoutes* that are not sub-paths of any other path in *SigRoutes* are returned, and the algorithm terminates. The purpose of returning significant routes that are not sub-paths of any other path is to improve solution quality. For example, if $\langle N_1, N_2 \rangle$ and $\langle N_1, N_2, N_3 \rangle$ are both found to be significant, only $\langle N_1, N_2, N_3 \rangle$ is returned.

Algorithm 2 Naïve Significant Route Miner (SRM_Naïve) Algorithm

Input:

1) A spatial network $G = (N, E)$ with a set of geo-referenced activities with point locations on network nodes or edges and weight function $w(u, v) > 0$ for each edge $e = (u, v) \in E$ (e.g., network distance),
2) A density ratio (λ) threshold, θ_λ,
3) A p-value threshold, θ_p,
4) m, indicating the number of Monte Carlo simulations,
5) A minimum distance (ϕ) threshold θ_ϕ

Output:

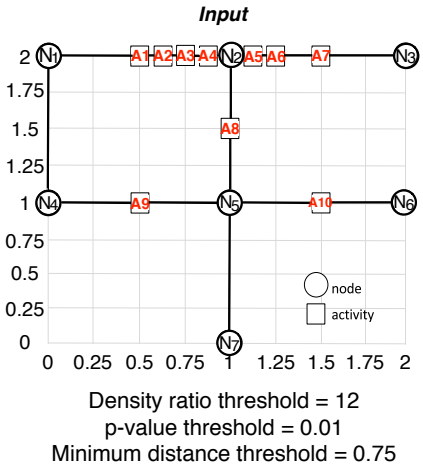
All routes $r \in R$ with $\lambda_r \geq \theta_\lambda$, $\phi_r \geq \theta_\phi$ and p-value significance level

Algorithm:

1: $G_{DS} \leftarrow$ dynamically segment G
2: **{Step 1:}** $P \leftarrow$ calculate all-pairs shortest paths in G_{DS}
3: **{Step 2:}** $Candidates \leftarrow$ paths in P having $\lambda \geq \theta_\lambda$ and $\phi \geq \theta_\phi$
4: **{Step 3:}** $SigRoutes \leftarrow$ significant paths in $Candidates$ using m Monte Carlo simulations
5: **{Step 4:} return** paths that are not sub-paths of any other path in *SigRoutes*

SRM_Naïve Example: Figure 2.4 shows an example execution trace of SRM_Naïve. The spatial network has 7 nodes, 7 edges, and 10 activities with specific locations on the edges. The density ratio threshold θ_λ is set to 12, the p-value threshold θ_p is set to 0.001, and the minimum distance threshold θ_ϕ is set to 0.75.

In step 1, all-pairs shortest paths in the given dynamically segmented spatial network are calculated (only paths with high density ratios are shown in the figure). For example, the shortest path between nodes A_1 and A_2 is $\langle A_1, A_2 \rangle$. In step 2, the density ratio, λ, for each shortest path is determined (see Definition 5) and paths whose $\lambda \geq \theta_\lambda$ and



Step 1 | Step 2 | Step 3

Start	End	Path	a_p	w_p	λ_p	p-value
A_1	A_7	<A_1,A_2,A_3,A_4 N_2,A_5,A_6,A_7>	7	1	14	0.01
A_1	A_6	<A_1,A_2,A_3,A_4 N_2,A_5,A_6>	6	0.75	12.5	0.01
A_1	A_2	<A_1,A_2>	2	0.125	13.75	0.01
A_2	A_3	<A_2,A_3>	2	0.125	13.75	0.01
A_3	A_4	<A_3,A_4>	2	0.125	13.75	0.01
A_5	A_6	<A_5,A_6>	2	0.125	13.75	0.01
...						

Output

Start	End	Path	λ_p	p-value
A_1	A_7	<A_1,A_2,A_3,A_4 N_2,A_5,A_6,A_7>	14	0.01

Figure 2.4: Example execution trace of Naïve Significant Route Miner (SRM_Naïve). Circles represent nodes and lines represent edges.

$\phi \geq \theta_\phi$ are stored as candidates. In the figure, from the 6 paths listed whose $\lambda \geq \theta_\lambda$, only the first two paths are considered as candidates since their distances meet or exceed the threshold. In step 3, the statistical significance of each candidate is calculated using Monte Carlo simulations (discussed next). Both of the 2 candidates meet the p-value threshold of 0.01. In step 4 (shown as the output), all paths among the significant paths that are not sub-paths of any other path are returned as significant routes. In this example, path $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7 \rangle$ are returned since it is the super-path of the other candidate. To reduce the prohibitive computational cost of SRM_Naïve, a new algorithm (SRM_GIS) is proposed in our previous work [86].

2.3.2 Significant Route Miner with Density Ratio Pruning and Monte Carlo Speedup (SRM_GIS)

The SRM_GIS algorithm uses filter and refine techniques (e.g., density ratio pruning and Monte Carlo speedup) to achieve computational savings. The filter and refine techniques may not change worst case complexity but they can reduce runtime in many cases. Density ratio pruning creates a boundary via an upper-bound density ratio such that not all destinations are visited from each source node. Some of the destinations

are pruned because the shortest paths to them will never meet the likelihood ratio threshold. Monte Carlo speedup avoids generating all shortest paths in cases where a shortest path in the simulated dataset has a higher density ratio than the shortest paths in the original dataset. Monte Carlo speedup also terminates early if the p-value threshold will not be met based on the number of times the maximum density ratio in the simulated dataset beats the maximum density ratio in the original dataset.

Density Ratio Pruning

Density ratio pruning aims to reduce the need to calculate all-pairs shortest paths in the graph G_{DS} based on the given density ratio threshold θ_λ. It is based on the idea that for each shortest path p, it is possible to determine an upper-bound density ratio for the super-paths rooted at p's start node, without calculating those super-paths.

The intuition behind the upper-bound density ratio for path p is that (1) the number of activities on all of p's super-paths rooted at p's start node are bounded by the number of activities in the spatial network minus the number of activities in the current shortest path tree rooted at the source node in p and (2) the weight of any super-path of p is at least the weight of the closest edge to p plus p's weight. Using the upper-bound density ratio makes it possible to terminate the all-pairs shortest paths computing earlier. However, since the dynamically segmented network varies during each Monte Carlo simulation trial, the shortest paths need to be computed in each trial. Even though each trial may be early terminated by the upper-bound pruning, the cost is still high. In addition, to ensure the correctness of the results, a post-processing step to verify if the paths with high density ratios are actual shortest path needs to be added to each simulation trial. More details of density ratio pruning are in our previous paper [86]. Due to the limited speedup, density ratio pruning is no longer used in the proposed approaches in this chapter.

Monte Carlo Speedup

Monte Carlo speedup aims to calculate the p-value without the need to consider all shortest paths in each simulated graph. The basic idea is that once a shortest path in the simulated graph is found to have a higher density ratio than the maximum density ratio in the original graph, the simulation immediately ends with the p-value being incremented. In other words, there is no reason to keep looking at all shortest paths in the simulated graph if we find one that already beats the maximum density ratio in the

original graph. Additionally, Monte Carlo speedup stops all simulations the moment p out of m simulations are found where the simulated density ratio beats the original maximum density ratio. In other words, there is no reason to execute all m simulations if we find that the p-value threshold will not be met.

Lemma 1 *Monte Carlo Speedup is a correct method for calculating p-value.*

Proof 1 *A method for calculating p-value is correct if it accurately determines the number of times, p, out of m simulations that the simulated density ratio beats the original maximum density ratio. Monte Carlo speedup only increments p when a density ratio is found in the simulated graph that beats the original density ratio. Thus, Monte Carlo Speedup is correct.*

2.4 Proposed Approach

In section 2.3, we review SRM_GIS [86] to address the LHDSP problem. However, the high computational cost makes it not capable to handle large amount of data. In this section, we propose a new algorithm (SRM_TBD) to significantly scale up the solution to the LHDSP problem. Two new algorithmic refinements, namely neighbor node filter algorithm and shortest path tree pruning algorithm (SPTP) are introduced, respectively.

2.4.1 Algorithms

Neighbor Node Filter

The idea behind the neighbor node filter is to obtain the shortest path between a pair of activities by stitching together a shortest path between two static nodes in the original network (i.e., the network before dynamic segmentation) with paths between the activities and the static nodes. Recall that in SRM_GIS [86], due to the activity shuffle, the dynamically segmented spatial network changes during each Monte Carlo simulation trial, which requires computing all-pairs shortest paths in each trial. In addition, since activities are considered as dynamic nodes in the dynamically segmented spatial networks, the cost for all-pairs shortest path computing gets higher than it would in the original network. Neighbor node filter reduces the computational cost in two ways. First, since the network does not change during Monte Carlo simulation trials, all-pairs

shortest paths only need to be computed once. Second, only shortest paths between static nodes need to be computed. This reduces the all-pairs shortest path computing cost compared with SRM_GIS.

The pseudocode for the neighbor node filter can be found in Algorithm 3. The algorithm first computes all-pairs shortest paths between static nodes of the original spatial network (line 1). Then, the algorithm seeks to avoid directly calculating all-pairs shortest paths between activities. Instead it determines these paths by stitching together shortest paths between static nodes in the original network P_{N_A} with paths between activities and the start and end of paths in P_{N_A} (lines 2∼5). To determine the exact shortest path for each pair of activities, it needs to find the shortest one from up to 4 stitched together paths. An illustrative example of this step is shown in next paragraph. In the Monte Carlo simulations, the original network never changes since only activities are shuffled. Therefore, only distances between each pair of activities need to be computed; there is no need to recompute all-pairs shortest paths (line 7).

The speedup of the neighbor node filter comes from two directions. First, instead of computing the all-pairs shortest paths on the dynamically segmented graph, it only computes the shortest paths between static nodes. Second, it avoids computing all-pairs shortest paths in each Monte Carlo simulation trial. In Section 2.4.2 and Section 2.6, it is evaluated theoretically and experimentally, respectively.

Neighbor Node Filter Example of Step 1: Figure 2.5 illustrates an example of the neighbor node filter. Shortest path $\langle A_1, A_{10}\rangle$ is calculated by selecting the shortest among 4 stitched together paths: (1) $\langle A_1, N_1\rangle$, $\langle N_1, N_5\rangle$, $\langle N_5, A_{10}\rangle$; (2) $\langle A_1, N_1\rangle$, $\langle N_1, N_6\rangle$, $\langle N_6, A_{10}\rangle$; (3) $\langle A_1, N_2\rangle$, $\langle N_2, N_5\rangle$, $\langle N_5, A_{10}\rangle$; and (4) $\langle A_1, N_2\rangle$, $\langle N_2, N_6\rangle$, $\langle N_6, A_{10}\rangle$. As the result, shortest path $\langle A_1, A_{10}\rangle$ is $\langle A_1, N_2\rangle$, $\langle N_2, N_5\rangle$, $\langle N_5, A_{10}\rangle$.

Shortest Path Tree Pruning A further refinement to the neighbor node filter is the shortest path tree pruning (SPTP). In the neighbor node filter, the search space of activity pairs is $4 \times a^2$, where a is the total number of activities. One way to reduce this search space is to eliminate certain activity pairs using an upper-bound pruning approach. Based on this idea, we propose the shortest path tree pruning algorithm (SPTP) which prunes a number of activities that are impossible to pair with for each activity. Given a spatial network, a set of activities, and a density ratio threshold, therefore, SPTP generates a list of candidate activities associated with each activity in

Algorithm 3 Significant Route Miner using the Neighbor Node Filter and Monte Carlo simulation speedup (SRM_NN)

Inputs and Outputs are the same as SRM_Naïve
Algorithm:
{Step 1: Calculate shortest paths between activities}
1: $P_{N_A} \leftarrow$ shortest paths between active nodes in G
2: **for each** $a_i \in A$ **do**
3: **for each** $a_j \in A$ **do**
4: $P \leftarrow shortestpath(x_i, x_j)$ based on combining shortest paths in P_{N_A} with the paths between each activity a_i and a_j and the nodes of their original edge
5: $P \leftarrow$ shortest paths between all pairs of activities
6: Step 2 the same as SRM_Naïve
7: Step 3 use Monte Carlo Speedup with reuse of P_{N_A} in each trial
8: Step 4 the same as SRM_Naïve

the network. After SPTP, instead of evaluating paths between all pairs of activities, we only need to evaluate the paths between an activity and its associated candidate activities.

The pseudocode for SPTP is shown in Algorithm 4. For each shortest path tree of the original graph G, SPTP does a depth-first-search with upper-bound pruning that eliminates those activities impossible to pair with any activity adjacent to the root of the tree. The search starts at the root (lines 3∼4), then traverses along the tree based on a depth first routine. For each path ($\langle N_{start}, N_{end} \rangle$) under search, an upper-bound of density ratio ($\overline{a}$) is computed using an upper-bound of activities and a lower-bound of weight ($\underline{w}$). Specifically, $\overline{a}$ is computed by adding 4 terms together. The first term is the number of activities on the path under search. The second term represents the number of activities on the edge that has the largest number of activities among all the edges adjacent to N_{start} except the edges in the path under search. The third term is the total number of activities on the subtree rooted at N_{end}. The fourth term represents the number of activities on the edge that has the largest number of activities among all the edges within or adjacent to the subtree rooted at N_{end} except the edges in the path under search. The lower-bound of weight $\underline{w}$ is the weight of path $\langle N_{start}, N_{end} \rangle$ (line 8). After that, the upper-bound of density ratio $\overline{\lambda}$ is computed (line 9) from $\overline{a}$ and $\underline{w}$. If $\overline{\lambda}$ is equal to or exceeds the density ratio threshold θ_λ, all the activities adjacent to

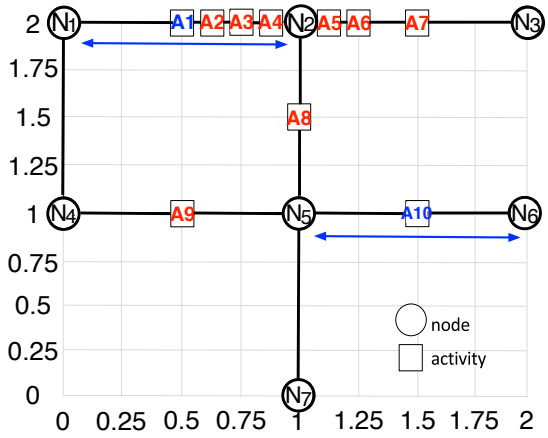


Figure 2.5: Example of Neighbor Node Filter for LHDSP. Shortest paths between activities are determined by stitching together (1) shortest paths between nodes in the statically segmented network with (2) paths between activities and the start and end of their original edges. Shortest path $\langle A_1, A_{10} \rangle$ is calculated by stitching together $\langle A_1, N_2 \rangle$, $\langle N_2, N_5 \rangle$, and $\langle N_5, A_{10} \rangle$ (Best in color).

N_{end} are put into the list of each activity adjacent to N_{start}, and the search continues on the subtree rooted at N_{end} (line 10∼13). Otherwise, the search ignores the subtree rooted at N_{end} which means that any of the activities adjacent or within it is impossible to pair with any activity adjacent to N_{start}. Note that the activities adjacent to the same active node, which means the path between them may not contain a path between static nodes, are always put in the list.

Shortest Path Tree Pruning Example: Figure 2.6 shows the shortest path tree rooted at N_1 from the graph in Figure 2.1. Note that the blue edge between N_4 and N_5 is an edge in the graph but not in the tree. Each edge is associated with two numbers, indicating its number of activities (red) and its weight (black). It picks the path having the largest number of activities as the shortest path if there are more than one shortest paths of equal weight between a pair of nodes. With the density ratio threshold set to 12, the algorithm starts from N_1, searching either edge $\langle N_1, N_2 \rangle$ or edge $\langle N_1, N_4 \rangle$ as the first step. For the purpose of illustration, we select $\langle N_1, N_2 \rangle$. Then, we compute the upper-bound of the density ratio $\overline{\lambda}$ of edge $\langle N_1, N_2 \rangle$ by computing $\overline{a}$ and $\underline{w}$. To compute $\overline{a}$ of edge $\langle N_1, N_2 \rangle$, we add four numbers: (1) the number of activities on $\langle N_1, N_2 \rangle$; (2) the number of activities on $\langle N_1, N_4 \rangle$ since it has the largest number of activities among

Algorithm 4 Shortest Path Tree Pruning Algorithm (SPTP)

Input:

1) A spatial network $G = (N, E)$ with a set of geo-referenced activities
2) Shortest path trees $T(N_i)$ rooted at active node N_i, $i = 1, \cdots, |n_a|$, where $|n_a|$ is the number of active nodes in G
3) A density ratio threshold θ_λ

Output:

A list of candidate activities $List_{a_j}$ associated with each activities a_j in G

Algorithm:

1: $List_{a_j} \leftarrow \emptyset$, $j = 1, \cdots, |A|$, Stack $s \leftarrow \emptyset$
2: **for each** $T(N_i)$ **do**
3: $N_{start} \leftarrow N_i$
4: $s \leftarrow$ Push (all children of N_{start})
5: **while** s is not empty **do**
6: $N_{end} \leftarrow$ Pop (s)
7: $\overline{a} \leftarrow$ number of activities on $\langle N_{start}, N_{end} \rangle$
+ number of activities on the edge among all the edges adjacent to N_{start} that has the largest number of activities except edges in $\langle N_{start}, N_{end} \rangle$
+ number of activities on $Subtree(N_{end})$
+ number of activities on the edge among all the edges adjacent to or within $Subtree(N_{end})$ that has the largest number of activities except edges in $\langle N_{start}, N_{end} \rangle$
8: $\underline{w} \leftarrow w_{\langle N_{start}, N_{end} \rangle}$
9: $\overline{\lambda} \leftarrow \frac{\overline{a} \div \underline{w}}{(|A| - \overline{a}) \div (|W| - \underline{w})}$
10: **if** $\overline{\lambda} \geq \theta_\lambda$ **then**
11: **for each** activities a_j adjacent to N_{start} **do**
12: $List_{a_j} \leftarrow List_{a_j}+$ all activities on all the adjacent edges of N_{end}
13: $s \leftarrow$ Push (all children of N_{end})
14: $N_{start} \leftarrow N_{end}$

all the edges adjacent to N_1 except $\langle N_1, N_2 \rangle$; (3) the number of activities on the subtree rooted at N_2; and (4) the number of activities on $\langle N_2, N_3 \rangle$ since it has the largest number of activities among all the edges within or adjacent to the subtree rooted at N_2 except $\langle N_1, N_2 \rangle$. Hence, $\overline{a} = 4 + 0 + 5 + 3 = 12$. The lower-bound of weight $\underline{w}$ is the weight of $\langle N_1, N_2 \rangle$, which is 1, therefore, $\overline{\lambda} = \infty$. Since ∞ exceeds the density ratio threshold (i.e., 12), the search continues and the 8 activities adjacent to N_2 are put into the list of each of the 4 activities adjacent to N_1. Next, The search continues on the path $\langle N_1, N_5 \rangle$. To compute $\overline{a}$ of path $\langle N_1, N_5 \rangle$, we add four numbers: (1) the

number of activities on $\langle N_1, N_5 \rangle$; (2) the number of activities on $\langle N_1, N_4 \rangle$ since it has the largest number of activities among all the edges adjacent to N_1 except $\langle N_1, N_2 \rangle$; (3) the number of activities on the subtree rooted at N_5; and (4) the number of activities on $\langle N_5, N_6 \rangle$ since it has the largest number of activities among all the edges within or adjacent to the subtree rooted at N_2 except $\langle N_1, N_5 \rangle$. Hence, $\overline{a} = 5 + 0 + 1 + 1 = 7$. The lower-bound of weight $\underline{w}$ is the weight of $\langle N_1, N_5 \rangle$, which is 2, therefore, $\overline{\lambda} = 5.83$. Since 5.83 is smaller than the density ratio threshold (i.e., 12), the search on this branch terminates and turns to path $\langle N_1, N_3 \rangle$. A list of activity pairs will be found during the depth-first search on the shortest path tree rooted at each active nodes.

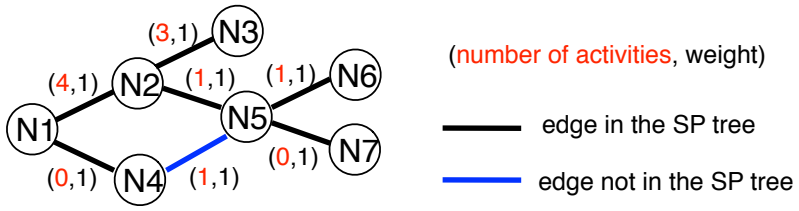


Figure 2.6: Example of Shortest Path Tree Pruning.

We propose a Significant Route Miner with both neighbor node filter, shortest path tree pruning, and Monte Carlo simulation speedup (SRM_TBD) whose pseudocode is shown in Algorithm 5. In step 1, Instead of traversing all activity pairs which is did in SRM_NN, SRM_TBD runs the SPTP algorithm to reduce the number of activity pairs. Step 2∼4 are the same as SRM_NN.

2.4.2 Theoretical Analysis

In this subsection, we first give some necessary lemmas that show the correctness and completeness of the proposed SRM_NN and SRM_TBD algorithms. Then, we analyze the time complexities of these algorithms, showing that in general cases, SRM_TBD is faster than SRM_Naïve and SRM_NN.

Lemma 2 *Dynamic Segmentation via Algorithm 1 does not lose any significant shortest path p between activities if p is not a subset of any other significant shortest path between the activities.*

Algorithm 5 Significant Route Miner with neighbor node filter, shortest path tree pruning, and Monte Carlo simulation speedup (SRM_TBD)

Inputs and Outputs are the same as SRM_Naïve
Algorithm:
{Step 1: Calculate shortest paths between activities}
1: $P_{N_A} \leftarrow$ shortest paths between active nodes in G
2: $List_{a_i} \leftarrow$ results from Shortest Path Tree Pruning (SPTP) algorithm, $i = 1, \cdots, |A|$
3: **for each** $a_i \in A$ **do**
4: **for each** $a_j \in List_{a_i}$ **do**
5: $P \leftarrow shortestpath(a_i, a_j)$ based on combining shortest paths in P_{N_A} with the paths between each activity a_i and a_j and the nodes of their original edge
6: $P \leftarrow$ shortest paths between all pairs of activities
7: Step 2∼4 the same as SRM_NN

Proof 2 *A method for adding nodes and edges to the spatial network based on the locations of activities is correct if it accurately adds each activity as a new node and inserts edges between all nodes. Since the location of each activity is required for dynamic segmentation, no activities are pruned, and all new nodes and edges are inserted (deleting original edges that have been segmented), dynamic segmentation via Algorithm 1 is correct.*

Lemma 3 *The shortest path enumerated in the neighbor node filter is correct.*

Proof 3 *This may be argued in a similar manner to that of hierarchical routing [95]. The key idea is that the shortest path between each activity must pass through either the start or end node of each original edge that the activity was on in the original spatial network before dynamic segmentation. Therefore, the overall shortest path can be enumerated by considering only the shortest path between these static nodes and stitching them to the shortest paths between each activity and the nodes of the original edge it was on. Since all the up to* 4 *paths between each pair of activities are considered, the neighbor node filter will find the correct path between any pair of activities.*

Lemma 4 *SRM_NN is complete and correct.*

Proof 4 *SRM_NN is complete if it can find all shortest paths that fulfill the density ratio threshold and the p-value threshold. The shortest path between each pair of activities*

is enumerated, and the density ratio and p-value evaluation find all enumerated paths that fulfill the density ratio and p-value thresholds. Therefore, SRM_NN is complete. SRM_NN is correct if all significant paths found by SRM_NN fulfill the density ratio threshold and the p-value threshold. According to Lemma 3, all enumerated paths between activities must be shortest paths. Density ratio and p-value evaluation eliminate all enumerated paths that do not fulfill the density ratio or p-value threshold. Therefore, SRM_NN is correct.

Lemma 5 *Shortest path tree pruning (SPTP) is correct pruning.*

Proof 5 *The correctness of SPTP means that the density ratio upper-bound $\overline{\lambda}$ of a path p is an overestimate of its exact density ratio λ. First, suppose shortest path p is stitched by three shortest paths: $\langle A_1, N_1\rangle$, $\langle N_1, N_2\rangle$, and $\langle N_2, A_2\rangle$, where A_1 and A_2 are the two ends of p, N_1 and N_2 are the static nodes next to A_1 and A_2, respectively. $\overline{\lambda}$ is computed from the shortest path tree rooted at either N_1 or N_2. Specifically, the number of activities on $\langle A_1, N_1\rangle$ is overestimated by the second term of $\overline{a}$ in SPTP. The number of activities on $\langle N_1, N_2\rangle$ is overestimated by the sum of the first and third terms of $\overline{a}$ in SPTP. The number of activities on $\langle N_2, A_2\rangle$ is overestimated by the fourth term of $\overline{a}$ in SPTP. Therefore, the number of activities upper-bound $\overline{a}$ of a path p is an overestimate of its exact number of activities a. Second, since the weight of $\langle N_1, N_2\rangle$ must be smaller than or equal to the sum weight of $\langle A_1, N_1\rangle$, $\langle N_1, N_2\rangle$, and $\langle N_2, A_2\rangle$, the lower-bound weight $\underline{w}$ of p is an underestimate of its exact weight. In addition, the case that A_1 and A_2 are adjacent to the same static node, the path $\langle A_1, A_2\rangle$ are always evaluated. In summary, SPTP is a correct pruning.*

Lemma 6 *SRM_TBD is complete and correct.*

Proof 6 *SRM_TBD is complete if SRM_TBD can find all shortest paths that fulfill the density ratio threshold and the p-value threshold. Since all paths that have higher density ratios than the threshold survive the SPTP (Lemma 5) and are sent to neighbor node filter. Since SRM_NN is complete, SRM_TBD is complete. SRM_TBD is correct means all significant paths found by SRM_TBD fulfill the density ratio threshold and the p-value threshold. Since all paths that survive the SPTP are sent to SRM_NN which is correct (Lemma 4), SRM_TBD is correct.*

Computational Cost Analysis:

The computational costs of SRM_Naïve, SRM_GIS and the proposed algorithms SRM_NN and SRM_TBD stem from 1) the cost of calculating all-pairs shortest path and 2) the cost of assessing statistical significance for all shortest paths in the spatial network.

Cost of SRM_Naïve and SRM_NN: the total cost of SRM_Naïve is $O((m+1) \times ((|N_S|+|N_D|)^2 log(|N_S|+|N_D|)+(|N_D|)^2)) = O(m \times ((|N_S|+|N_D|)^2 log(|N_S|+|N_D|)))$, where $|N_S|$ is the number of static nodes, and $|N_D|$ is the number of dynamic nodes. $O((|N_S|+|N_D|)^2 log(|N_S|+|N_D|))$ is the cost of calculating all-pairs shortest paths in the spatial network, $O(|N_D|^2)$ is the cost of calculating the density ratios of all enumerated paths between activities, and m is the number of Monte Carlo simulations. Note that the cost of all-pairs shortest path computing is $O((|N_S|+|N_D|)|E|+(|N_S|+|N_D|)^2 log(|N_S|+|N_D|))$ with a priority queue implemented by Fibonacci heap, where $O(|E|)$ indicates the number of edges in the network. Since transportation networks are sparse so that $O(|E|)$ is dominated by $O((|N_S|+|N_D|)log(|N_S|+|N_D|))$, we simplify this cost to $O((|N_S|+|N_D|)^2 log(|N_S|+|N_D|))$ in our analysis. In addition, the cost of computing the density ratio of a path is $O(1)$ since the number of activities and weight of each shortest path can be stored during the all-pairs shortest paths computing.

Cost of SRM_NN: The total cost for SRM_NN is $O(|N_S|^2 log|N_S| + f_{MC} \times m \times (|N_D|+|N_S| \times |E|+|N_D|^2))$, where $O(|N_S|^2 log|N_S|)$ is the cost for computing all-pairs shortest paths on the original spatial network, $O(m \times (|N_D|+|N_S| \times |E|+|N_D|^2)$ indicates the cost of calculating the density ratios of all enumerated paths between activities. Specifically, it costs $O(|N_D|)$ to find out the number of activities on each edge. Then, for each shortest path tree rooted at a static node, it costs $O(|E|)$ to find the number of activities for each shortest path in the tree by accumulating the numbers of activities following a "root-to-leaf" routine. In addition, f_{MC} is an indicator of the speedup from Monte Carlo speedup the same as in SRM_GIS. The cost for calculating the density ratios will be simplified to $O(m \times |N_D|^2)$ which are the dominating terms.

It can be seen that the speedup over SRM_GIS comes from the all-pairs shortest path computing. First, in SRM_NN, all-pairs shortest paths need to be computed only once instead of being computed in each Monte Carlo simulation trial as in SRM_GIS. This decreases the cost a lot since it typically needs $100 \sim 1000$ trials in the Monte

Carlo simulation. In addition, number of nodes taken into consideration decreases from $|N_S|+|N_D|$ to $|N_S|$. This difference can be large since $|N_D|$ could be big when the time period of the dataset is long.

Cost of SRM_TBD: The total cost for SRM_TBD is $O(|N_S|^2 log|N_S| + f_{MC} \times m \times (f_{DFS} \times |N_D|(|N_D| + |N_S|) + f_{SP} \times |N_D|^2))$. Compared with SRM_NN, it aims to reduce the cost of the path evaluation part. In each Monte Carlo simulation trial, depth first searches are run on N_D shortest path trees, each costing $O(|N_D| + |N_S|)$. However, the depth first searches are not necessarily complete with existence of the pruning, therefore, f_{DFS} is multiplied to indicate the speedup from the pruning, where $0 \leq f_{DFS} \leq 1$. Moreover, the path evaluation does not need to go through all activity pairs, f_{SP} shows a speedup from that, where $0 \leq f_{SP} \leq 1$. In worst case, SRM_TBD costs more than SRM_NN since the depth first search takes some time but may not prune anything. However, in practice, the pruning ratio is always high, giving SRM_TBD a significant speedup over SRM_NN (shown in Section 2.6).

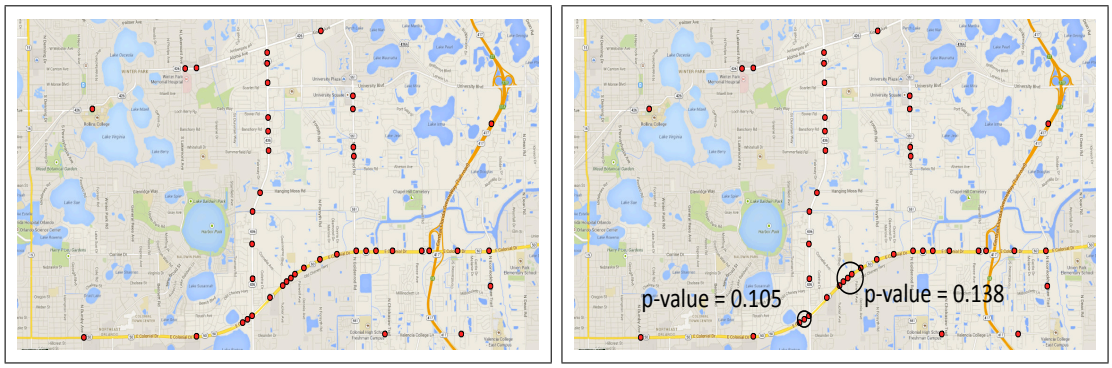
2.5 Case studies

We conducted two qualitative evaluations of our algorithm (SRM_TBD) with and without dynamic segmentation by comparing their results with the results of SaTScan [69] on two real data sets. First, we used a real pedestrian fatality data set [3], shown in Figure 2.7(a). As noted earlier, SaTScan discovers areas of significant activity that are represented as circles on the spatial network while SRM_TBD discovers significant shortest paths. The input consisted of 43 pedestrian fatalities (represented as dots) in Orlando, Florida occurring between 2000 and 2009. For each edge (portion of road) in the network, fatality count was aggregated, yielding an overall activity number. Weights of edges were the actual road network distance. The maps were prepared using QGIS' Open Layers plugin [96], and the network was from the US Census Bureau's TIGER/Line Shapefiles [97].

To evaluate the techniques, we considered the outputs of circles vs. shortest paths. We used a p-value threshold of 0.04 for our linear hotspot discovery approach. As noted earlier, pedestrian fatalities usually occur on streets, particularly along arterial roadways [1]. Thus this activity can be said to have a linear generator. However, the results

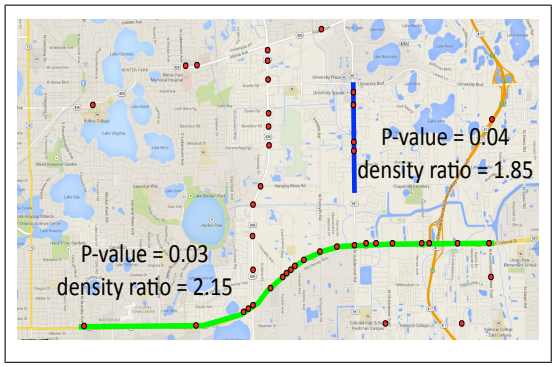
with high p-values generated by SaTScan do not capture this. From Figure 2.7(b), it is clear that SatScan's circle-based output is meant for areas, not streets. In contrast, the shortest paths detected by SRM_TBD without dynamic segmentation fully capture the significant activities on the arterial roads (some of the paths in Figure 2.7(c) are overlapping). Furthermore, the paths in the figure make sense in this context due to the inherently linear nature of the activities.

We further compared SRM_TBD outputs generated with and without dynamic segmentation. With dynamic segmentation (Figure 2.7(d)), SRM_TBD is able to detect the linear hotspot marked as brown (on the road running vertically in the middle of the figure) which is completely missed without dynamic segmentation. The reason is that the overall density ratio of this vertical road is compromised by the empty portion at the bottom, making it not qualified as a hotspot. However, with dynamic segmentation, the dense portion of that road is found as a hotspot while the empty portion at the bottom is dismissed. This contrast shows that dynamic segmentation assists in discovering statistically significant hotspots that are previously missed. In addition, with dynamic segmentation, SRM_TBD gives hotspots with more accurate and precise locations and higher density ratios. For example the blue hotspots in Figure 2.7(c) and Figure 2.7(d) indicate the same road. However, the hotspot with dynamic segmentation indicates the precise location where the activities are clustered while the hotspot without dynamic segmentation contains empty portions at both the top and bottom end. Quantitatively, with dynamic segmentation, the density ratio of the blue hotspot is higher (i.e., 2.77 vs. 1.85) and the p-value is lower (i.e., 0.02 vs. 0.04). These results demonstrate that dynamic segmentation assists in discovering the location of hotspots more precisely than the other technique.

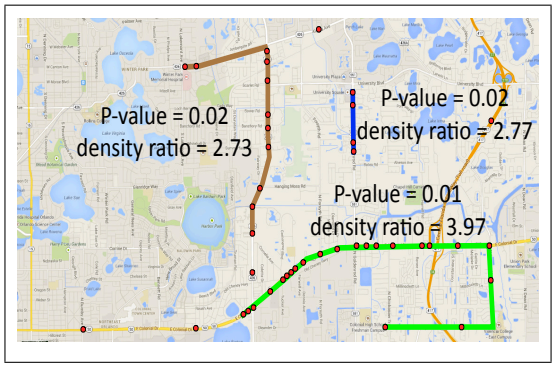


(a) Input

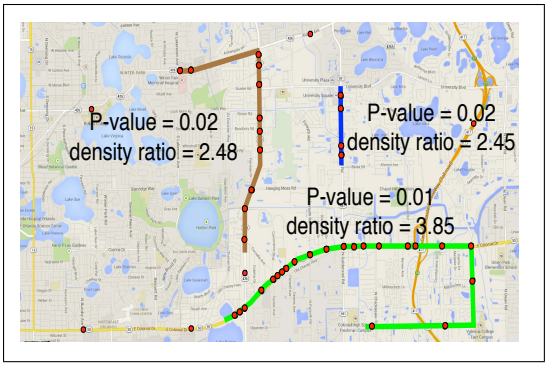
(b) SaTScan



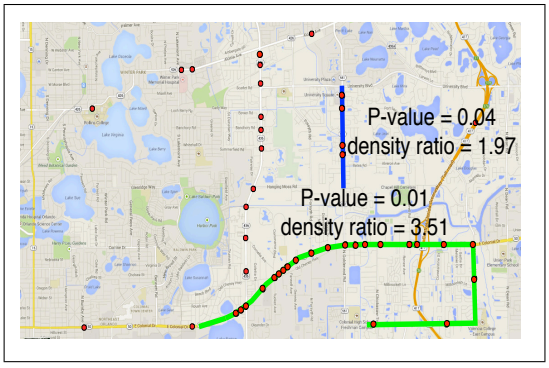
(c) SRM_TBD without dynamic segmentation



(d) SRM_TBD with dynamic segmentation



(e) SRM_TBD with road segments partitioned by fixed interval of 0.2km

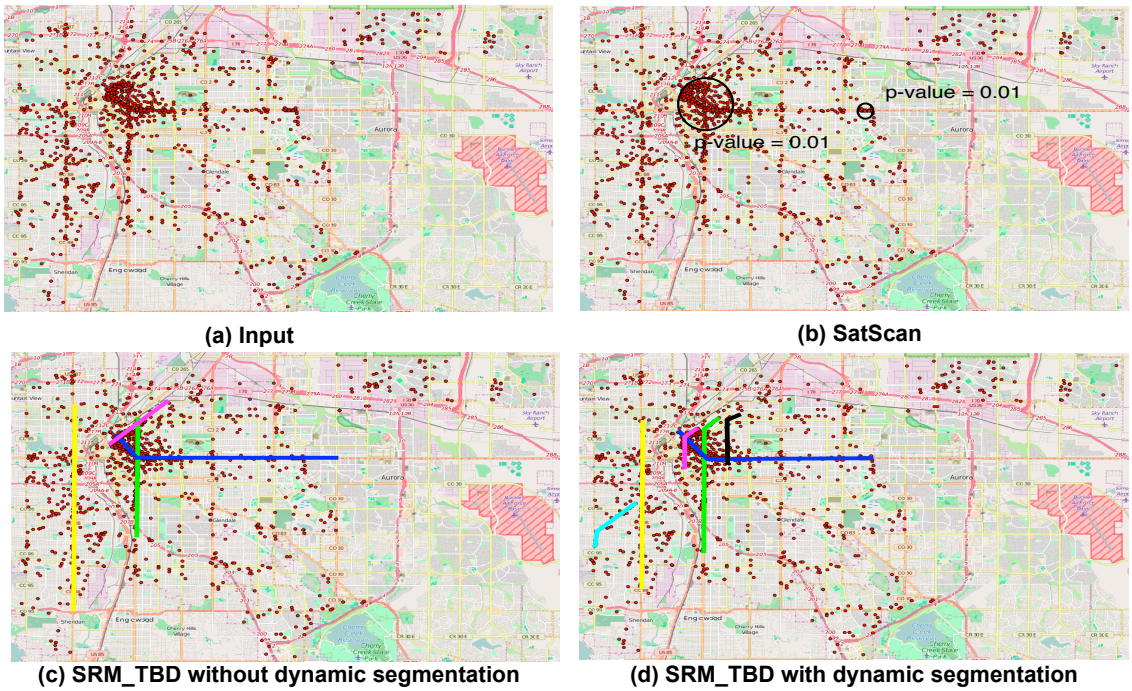


(f) SRM_TBD with road segments partitioned by fixed interval of 0.8km

Figure 2.7: Comparing SRM_TBD (without dynamic segmentation), SRM_TBD (without dynamic segmentation, roads are partitioned by fixed intervals), SRM_TBD (with dynamic segmentation), and SaTScan's output on pedestrian fatality data from Orlando, FL [3] (Best in color).

Also, long road segments in this network are partitioned with fixed intervals of 0.2km and 0.8km respectively, and then ran SRM_TBD without dynamic segmentation. The results detected using a small interval of 0.2km shown in Figure 2.7(e) are close to the results detected with dynamic segmentation shown in Figure 2.7(d) with minor difference at the ends of the hotspots, making the density ratios slightly smaller. Figure 2.7(f) shows the results detected using a large interval of 0.8km, in which the vertical hotspot (brown color) is missed. It can be seen that the results are subject to the length of the interval which is user-specified. If the interval is large, this method may still miss some patterns compared to dynamic segmentation; if the interval is small, this method may have comparable solution quality to dynamic segmentation, however, the computational cost will be expensive since the number of nodes in the partitioned network is much larger than in the original network.

In order to evaluate the SRM_TBD in larger datasets, we also conducted a case study on a dataset of 1529 simple assaults in Denver, Colorado during 2015 [98] with a p-value threshold of 0.01. As shown in Figure 2.8(b), SatScan finds a big circular hotspot that covers the downtown area as well as another small hotspot in the middle of the study area. Using a minimum distance threshold of 1km, SRM_TBD without dynamic segmentation discovers 4 nonoverlapped hotspots on the major roads shown in Figure 2.8(c). SRM_TBD with dynamic segmentation discovers 6 nonoverlapped hotspots shown in Figure 2.8(d). Some of them (e.g., yellow, green, blue) indicate the same roads as those without dynamic segmentation but capture slightly different portions of the roads, giving higher density ratios. In addition, some previously missed hotspots are captured using dynamic segmentation such as the cyan, pink, and black hotspots. The p-values and density ratios of the results are listed in the right side of Figure 2.8. Case study on this larger dataset also demonstrates that dynamic segmentation helps locate the hotspots more precisely and find previously missed hotspots.



(a) Input

(b) SatScan

(c) SRM_TBD without dynamic segmentation

(d) SRM_TBD with dynamic segmentation

P-values and density ratios of results without dynamic segmentation

Hotspot	P-value	Density ratio
Yellow	0.01	25.55
Green	0.01	29.68
Blue	0.01	28.76
Pink	0.01	23.18

P-values and density ratios of results with dynamic segmentation

Hotspot	P-value	Density ratio
Pink	0.01	26.42
Black	0.01	26.97
Yellow	0.01	30.17
Cyan	0.01	25.41
Green	0.01	32.94
Blue	0.01	33.52

Figure 2.8: Comparing SRM_TBD (without dynamic segmentation), SRM_TBD (with dynamic segmentation), and SaTScan's output on street robbery data from Denver, CO [3] (Best in color).

2.6 Experimental Evaluation

The goal of our experiments was to evaluate the scalability of the proposed approach in sub-edge level linear hotspot discovery. To achieve this goal, we designed two sets of experiments. First, we conducted self-analysis experiments that evaluate the effect of each algorithmic refinement under a varying number of activities in the dataset. Second, we conducted comparative analysis experiments that compare SRM_TBD with the baseline approaches (i.e., SRM_Naïve and SRM_NN).

2.6.1 Experiment Setup

Our experiments were performed on a real-world road network dataset obtained from the US Census Bureau's TIGER/Line Shapefiles [97] that contained about 500 nodes and 1000 edges. The weight of each edge was the actual road network distance. The varying number of activities used in the experiments were synthetic data generated under the complete spatial randomness. Network size was varied by putting virtual nodes on the edges. All experiments were performed on a Macbook Pro with an Intel Core i7 Quad Core 2.2 GHz processor and 16 GB RAM.

2.6.2 Effect of Algorithmic Refinements

Effect of the Neighbor Node Filter: The experiment to evaluate the neighbor node filter had two parts. The first part was designed to test how much speedup is earned without Monte Carlo (MC) simulations. We compared the running time between SRM_Naïve and SRM_Naïve with the neighbor node filter in one run of the all-pair shortest paths computing and density ratio evaluation with varying number of activities. The density ratio threshold was set to 5. Figure 2.9(a) shows the execution times in log scale. We found that the cost of SRM_Naïve grows much larger as the number of activities increases while the cost remains steady with the neighbor node filter. The reason is that the cost of all-pairs shortest paths computing scales up with the number of nodes in the dynamically segmented graph, which increases as the number of activities increases. In contrast, with the neighbor node filter, all-pairs shortest paths are only computed on the original graph, whose size does not vary with the number of activities. The slight increase of cost comes from the density ratio evaluation. The second part

of the experiment aimed to evaluate the speedup coming from the MC simulations. The parameters was set the same as the first part except 100 MC simulation trials are run with a p-value threshold set to 0.05. Figure 2.9(b) shows the execution times in log scale. We found that the speedup of neighbor node filter increases from 10 to 100 times as the number of activities increases. This speedup comes from both the all-pairs shortest paths computing as shown in the first part and the re-use of the shortest paths.

Effect of Shortest Path Tree Pruning: This experiment aimed to evaluate the speedup earned from the shortest path tree pruning (SPTP) approach. We compared the neighbor node filter algorithm to the approach using both neighbor node filter and SPTP. The density ratio threshold was set to 5, while the p-value threshold was set to 0.05 and the number of MC simulation trials was set to 100. Figure 2.9(c) shows the execution times in log scale. We found that the cost with SPTP grows more slowly as the number of activities increases.

Effect of Monte Carlo Simulation Speedup: We evaluated the speedup earned from the MC simulation speedup by comparing SRM_TBD's performance with and without Monte Carlo simulation speedup. The density ratio threshold was set to 5, while the p-value threshold was set to 0.05. Figure 2.9(d) shows the execution times in linear scale. We found that the speedup provided from MC speedup keeps about 20%.

Memory Cost Test: We evaluated the memory cost of SRM_NN and SRM_TBD under different sizes of network and number of activities, respectively. Figure 2.9(e) shows the memory costs with numbers of nodes varied from 250 to 2000, and Figure 2.9(f) shows the memory costs with numbers of activities varied from 200 to 1600. We found that SRM_TBD cost memory as around two times as SRM_NN. The main reason is that SRM_NN maintains only the all-pairs shortest paths but SRM_TBD also maintains the shortest path trees. We found that as the size of the network increased, the memory cost for both algorithms increased relatively fast since the total size of paths and trees increased quadratically. As the number of activities increased, the memory cost for both algorithms increased relatively slow since the total size of paths and trees increased linearly caused by the increase of number of active nodes.

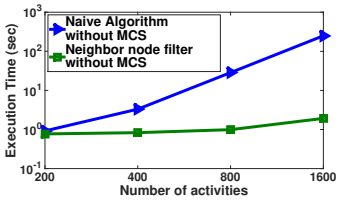
2.6.3 Results of Comparative-analysis Experiments

Effect of the Number of Activities: This experiment was designed to compare the performance of all approaches with varying numbers of activities among SRM_Naïve, SRM_GIS and SRM_TBD. The density ratio threshold was set to 5, p-value threshold was set to 0.05 and the number of Monte Carlo simulation trials was set to 100. We varied the number of activities from 200 to 1600. When the number of activities is 1600, dynamic segmentation increases the network size by adding 1600 new nodes to the 500 nodes in the original network. Figure 2.9(g) gives the execution times in log scale. SRM_TBD resides apart from SRM_Naïve and SRM_GIS, especially when the number of activities is larger. For example, when number of activities is 1600, SRM_TBD was about 100 times faster than SRM_GIS.

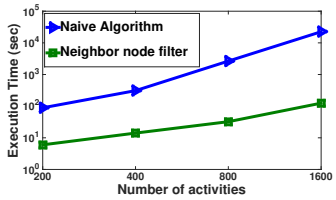
In addition, in order to test the scalability of the proposed algorithms in larger datasets, we varied the number of activities from 4000 to 32000 on a network containing 2000 nodes and 2500 edges. The new network was generated by putting virtual nodes on the edges of the orginal network. It ran only SRM_NN, SRM_NN with shortest path tree pruning (SPTP) and SRM_TBD on these larger datasets since SRM_Naïve and SRM_GIS cost prohibitive time (e.g., SRM_Naïve cost approximate 50000 seconds for 2000 activities). The execution times in log scale given Figure 2.9(h) show that the proposed algorithmic refinements maintained the performance on these datasets and SRM_TBD was able to handle larger datasets within reasonable running time.

Effect of the Density Ratio Threshold: In this experiment, the number of activities was set to 800, the p-value threshold was set to 0.05, and the number of MC simulations was set to 100. We varied the density ratio threshold from 2.5 to 10. The results are shown in Figure 2.9(i). As can be observed, SRM_TBD cost fewer when the density ratio threshold got larger while maintaining the speedup over SRM_Naïve and SRM_GIS.

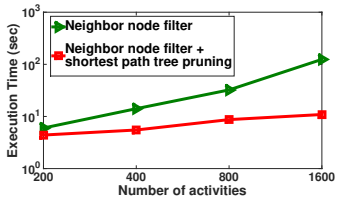
Effect of the Size of the Network: In this experiment, number of activities was set to 800, p-value threshold was set to 0.05, number of MC simulations was set to 100, and density ratio threshold was set to 5. We varied the number of nodes in the network from 250 to 500, while keeping the ratio between number of nodes and edges. This was realized by adding or removing virtual nodes on the existing edges of the real dataset. The results are shown in Figure 2.9(j). As observed, the cost of SRM_TBD resides apart from SRM_Naïve and SRM_TBD.



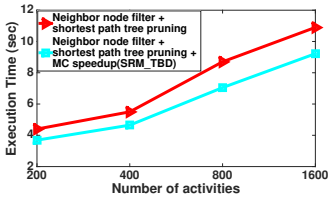
(a) Effect of neighbor node filter in shortest path computing



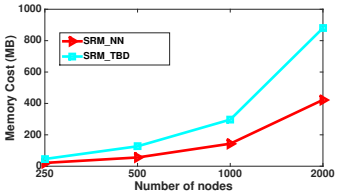
(b) Effect of neighbor node filter



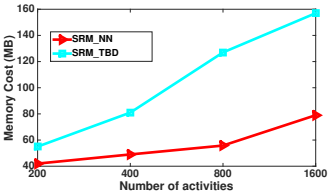
(c) Effect of shortest path tree pruning



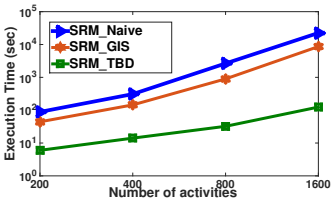
(d) Effect of Monte Carlo simulations



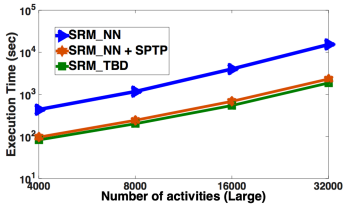
(e) Memory Cost with different sizes of network



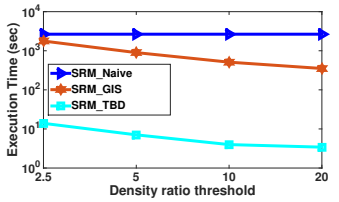
(f) Memory Cost with different numbers of activities



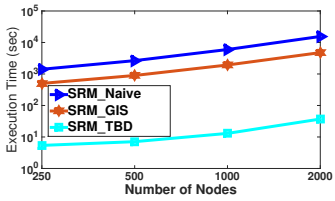
(g) Running time with different numbers of activities



(h) Running time with different numbers of activities (large)



(i) Running time with different Density ratio thresholds θ_λ



(j) Running time with different sizes of the network

Figure 2.9: Experimental evaluations of effect of each algorithmic refinement (Figure 2.9(a)-(f)) and comparisons among candidate algorithms under different varying parameters (Figure 2.9(g)-(j))

In summary, each of the three algorithmic refinements, namely the neighbor node filter, the shortest path tree pruning, and the Monte Carlo speedup, contributes to reduce the computational cost. The comparative-analysis experiments show that SRM_TBD performs much better than the SRM_Naïve approach and SRM_GIS [86] under the varying of different parameters of the dataset.

2.7 Discussion

Techniques without significance testing: This chapter focuses on partitioning techniques that consider statistical significance. There are a myriad of other techniques that divide data into groups without considering statistical significance. These include DBScan [89], K-Means [90], KMR [91], and Maximum Subgraph Finding [92]. For example, the algorithm from our previous work [91] on summarizing activities using routes may return routes that are not statistically significant. DBScan [89] finds clusters in Euclidean space. However, it is sensitive to the parameter selection and may return chance clusters even on a complete spatial randomness dataset. Post-processing the output of these techniques for statistical significance will not guarantee completeness as some of the clusters returned may not be statistically significant. We will explore ways to include statistical significance testing with traditional methods such as K-Means, etc.

Techniques to reduce memory cost: The proposed algorithm needs to store information of all-pairs shortest paths and shortest path trees, which takes a memory cost of $|N_S|^2$ where $|N_S|$ is the number of road intersections in the spatial network. When the size of the spatial network is large to a point, it not feasible to store all the information in memory. To deal with this problem, two techniques which trade off memory cost with running time can be applied. (1) Store the paths and trees in hard drive instead of memory. In this technique, the specific paths and trees are loaded into memory as they are acquired. This reduces the memory cost but increases I/O running time. However, this cost can be mitigated if the information is stored in a well-indexed structure. (2) Use a hierarchical structure. With a hierarchical structure, the spatial network is partitioned into multiple fragments. Only information within each fragment is stored in memory. Suppose the spatial network is partitioned into p fragments each having $|N_S|/p$ road intersections. The memory cost of storing shortest paths reduces

from $|N_S|^2$ to $(|N_S|/p)^2 \times p = |N_S|^2/p$ plus the cost for storing the boundary graph. However, when information across different fragments is acquired, on-time computation is needed which increase the running time. Detailed discussion of hierarchical structure applied in routing algorithms is available in related literature [99].

2.8 Conclusions and Future Work

This chapter explored the problem of significant linear hotspot discovery in relation to important application domains such as preventing pedestrian fatalities and crime analysis. It proposed algorithms that discovers multiple statistically significant shortest paths at the sub-edge level in a spatial network. The proposed approach uses a neighbor node filter, shortest path tree pruning, and Monte Carlo speedup to enhance its performance and scalability. Two case studies on pedestrian fatality and crime data validate the superiority of our approach over SatScan for detecting statistically significant hotspots of a linear nature. Experimental evaluation using real-world and synthetic data indicated that the algorithmic refinements utilized by our approach yield substantial computational savings without sacrificing result quality.

In future work, we plan to explore other types of data that may not be associated with a point in a street (e.g., aggregated pedestrian fatality data at the zip code level). The present research is centered on finding high concentrations of activities whose counts and locations are deterministic. However, future work is needed to investigate attributes that may not be deterministic such as delay when moving between nodes, capacity constraints, etc. Additionally, estimating p-values via Monte Carlo simulations may be done in different ways. The current approach permutes the activities. Alternatively, it can permute activity count which may provide a Poisson distribution assumption. In addition, the proposed problem aims to find significant path that are shortest paths. We plan to investigate how to define conceptually more meaningful paths than shortest paths and develop corresponding scalable algorithms. Finally, finding spatio-temporal linear hotspots is also a direction of our future work. Currently, we use an aggregated number of activities over time. Instead, in the future, we plan to find hotspots that incorporate temporal information [100, 101]. They are potentially applied to find life-cycle and moving trends of hotspots.

Chapter 3

Linear Hotspot Discovery on All Simple Paths

3.1 Introduction

Spatial hotspot discovery finds regions of interest which have statistically significant concentration of activities (e.g., traffic accidents). It is been applied in many societal applications including transportation engineering, public health, and public safety over the years. This chapter studies the problem of Linear Hotspot Discovery on All Simple Paths (LHDA) which identifies all the simple paths (i.e. paths with no loops) enumerated from a given spatial network (e.g., road network) that have statistically significant density of activities .

3.1.1 Application Domains

Improving pedestrian safety is a critical task in transportation engineering. One major cause of traffic accidents involving pedestrians is lack of proper infrastructure. For example, Figure 3.1(a) shows pedestrians walking in a motor vehicle lane since the sidewalk is covered by snow [1]. In contrast, as shown in Figure 3.1(b), Queens Boulevard in New York City once called the "Boulevard of Death", now has become dramatically safer because of newly built dividers between pedestrians, bicycles, and motorcycles [7]. Before allocating limited resources for improving infrastructure for the most needing

roads, LHDA provides an efficient data-driven approach for identifying accident-prone roads in cities which need urgent attention.

(a) (b)

Figure 3.1: (a) Pedestrians at risk on road [1]. (b) Queens Boulevard with new paved sidewalk and road separators [7].

In public health and epidemiology, identifying disease hotspots is critical for disease prevention, preparation and reduction [102]. Distinct from traditional hotspot detection approaches which assume isotropic diffusion of disease over Euclidean space [69], LHDA helps identify novel linear hotspots formed by the diffusion of disease via carriers traveling along roads in a transportation network or polluted water flowing down the length of a river. As an example, the region surrounding lower Mississippi River is identified as a hotspot of high incidence of colon-rectum cancer [102].

In public safety, there is a critical need to identify streets with high crime rates so the police force can be efficiently deployed [103]. LHDA helps identify those streets prone to high crime rates which cannot be easily discovered by the human eye due to the complexity of street structures and non-trivial distribution of crimes.

3.1.2 Challenges

The computational challenges of LHDA come from the following aspects. First, the size of studied spatial networks can be very large. For example, the number of road intersections in an average metropolitan area in the US can be millions. And, the number of hotspots can be exponential in the worst case since a complete graph containing $|N|$

nodes can generate $|N|!$ distinct simple path.

Moreover, dynamic programming (DP) approaches solving the Travelling Salesman Problem (TSP) [104] do not apply in LHDA. Those DP methods reuse computed optimal paths on subsets of the input nodes. However, a simple path with maximum density may not contain sub-paths with maximum density, which violates the sub-optimality assumed in the DP approach. For example, even though path $p_1 = (S, A, C, D)$ has higher density than path $p_2 = (S, B, C, E)$, the subpath (S, A, C) of p_1 can have a smaller density than subpath (S, B, C) of p_2.

Furthermore, existing acceleration approaches for solving shortest paths problem [105, 106] do not apply to LHDA. These methods shrink the search space by either pruning paths that are known to be far from the shortest paths [105] or by condensing the paths to a set of representative paths [106]. However, since we need to consider the complete set of simple paths, this type of path pruning and condensing based on the network structure do not work for LHDA.

3.1.3 Related Work

Traditionally, clustering algorithms (e.g., DBScan [89]) are used for detecting regions with high activity concentration. However, without statistical significance test, they tend to output false positive hotspots formed just by chance. False positives are unacceptable in many societal applications due to the potential severe consequences. For example, a neighborhood wrongly designated as a crime hotspot may experience unnecessary distress, lost business, or see its property values drop.

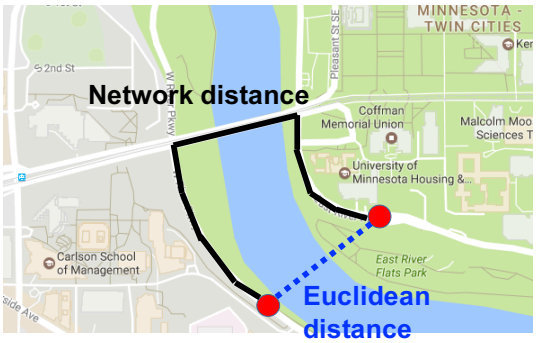


Figure 3.2: Network distance vs. Euclidean distance (best in color)

To eliminate the false positives, approaches with statistical significance test have been developed to detect hotspots modeled in Euclidean space. They discover hotspots in a collection of regular shapes such as circle, ellipse [69], and rectangle [84]. However, Euclidean space is inaccurate and biased [75] in modeling human activities that occur along roads (e.g., traffic accidents). Figure 3.2 shows two river banks connected by a bridge. The Euclidean distance between the two red dots is short (dashed line), while the network distance is much longer (solid line). When modeling activities that occur on the path between these two dots, the error using Euclidean distance will be huge.

More recent work has focused on spatial network models [76, 75]. Network iso-distance hotspot detection [76] discovers hotspots formed under the assumption that activities are isotropically diffused along road networks. However, in many cases, the shape of a hotspot is a linear path (e.g., Queens Blvd [7]), rather than an iso-distance sub-network. Another approach, Linear Hotspot Detection on Shortest Paths (LHDSP) [75], is able to discover linear hotspots. However, it only enumerates shortest paths as candidates due to the trade off between solution completeness and computational tractability. Thus, patterns in many real-world scenarios are missed since people often do not travel along shortest paths (e.g., commute route, bus route). Our proposed approach overcomes the limitations of related works by detecting linear hotspots enumerated from all simple paths (ASPs) on spatial networks.

3.1.4 Contribution

This chapter formulates the problem of Linear Hotspot Discovery on All Simple Paths (LHDA) which detects novel hotspot patterns which have not been studied before. To address the computational challenges in LHDA, we propose a novel prune-and-refine algorithm based on bi-directional fragment-multi-graph traversal (ASP_FMGT). Extensive theoretical and experimental evaluations show ASP_FMGT achieves substantial improvement in scalability over the baseline approach. Case studies on real-world datasets show that the proposed approach is able to locate linear hotspots with higher accuracy and discover patterns that are completely missed by related approaches.

3.1.5 Scope and Organization of This Chapter

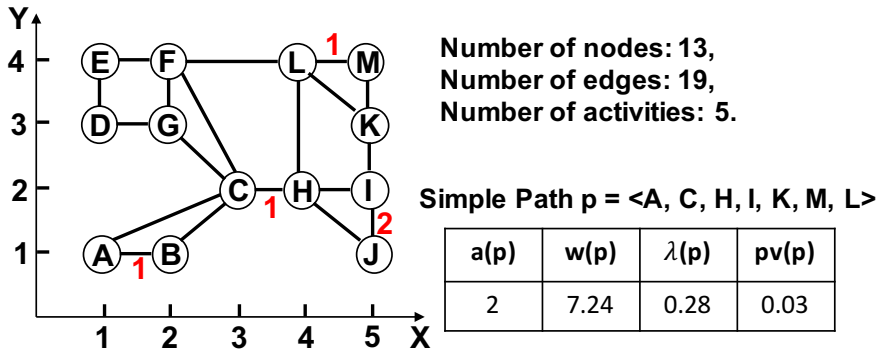
Our focus is discrete activities modeled as points on a spatial network. Modeling of other types of activities such as trajectories that represent objects moving over time is not considered. In addition, a thorough analysis of the underlying incentives of hotspots can only be done by domain experts and is out of the scope of this chapter.

The rest of this chapter is organized as follows: Section 3.2 presents the basic concepts and formulates the problem of LHDA. Section 3.3 and Section 3.4 propose a baseline algorithm ASP_Base and a novel algorithm ASP_FMGT, respectively. Section 3.5 analyzes the asymptotic time and space complexities. Section 3.6 discusses the experimental evaluation of the computational performance of the proposed approach. Section 3.7 presents two case studies using real-world datasets which evaluated the effectiveness of ASP_FMGT in hotspot detection. Section 3.8 concludes the chapter and previews future work.

3.2 Problem Statement

This section formulates the problem of Linear Hotspot Discovery on All Simple Paths (LHDA). It starts by introducing the necessary basic concepts then gives the formal statement of LHDA.

3.2.1 Basic Concepts



a(p)	w(p)	λ(p)	pv(p)
2	7.24	0.28	0.03

Figure 3.3: Illustration of LHDA (best in color)

Definition 9 *A **spatial network** $G = (N, E)$ consists of a node set N and an edge set E, where each node $n \in N$ is associated with a 2-D location $n = (x, y)$. Each edge $e = (n_i, n_j) \in E$ is an edge that connects node n_i to node n_j.*

The most typical spatial network is a road network where nodes represent street intersections and edges represent road segments. Figure 3.3 shows a spatial network that contains 13 nodes (i.e. A...L) and 19 edges. Note that even though the example network is undirected for easy illustration, our proposed approach can deal with both undirected and directed networks.

Definition 10 *A **simple path** is a sequence of k nodes $p = \langle n_1 \dots n_k \rangle$ with no repeated nodes (i.e. no loops), and two consecutive nodes n_i and n_j are connected by the edge between them (n_i, n_j).*

In Figure 3.3, Path $\langle A, C, H, I, K, M, L \rangle$ is a simple path, but Path $\langle A, C, H, I, J, H, L \rangle$ is not a simple path since it passes node H twice.

Definition 11 *An **activity set** A is a collection of activities, and each activity $a \in A$ is associated with an edge $e \in E$.*

In Figure 3.3, there are 5 activities (represented as the number associated with each edge) associated with 4 different edges (A, B), (C, H), (I, J), and (L, M).

Definition 12 *The **activity coverage** and **weight** of path p: $a(p)$ and $w(p)$ indicate the number of activities on p and the weight of p, respectively.*

Weight can be defined using varied metrics such as travel time, travel distance, and traffic flow. In Figure 3.3, activity coverage is represented as the red number along with each edge, and weight is represented by the length of the path. For example, Path $\langle A, C, H, I, K, M, L \rangle$ has an activity coverage of 2, and a weight of 7.24.

Definition 13 *The **density of path** p, $\lambda(p) = \frac{a(p)}{w(p)}$.*

In Figure 3.3, the density of Path $p = \langle A, C, H, I, K, M, L \rangle$ is $\frac{2}{7.24} = 0.28$. We use density as the metric for two reasons. First, it is a popular metric for crash rates used in transportation analysis [107]. Second, it has a mathematical property that provides opportunities for reducing computational cost.

Lemma 7 *Given a path p concatenated by k mutually exclusive subpaths $\{p_i^{sub}, i = 1 \dots k\}$, $\min\{\lambda(p_i^{sub})\} \leq \lambda(p) \leq \max\{\lambda(p_i^{sub})\}$.*

Proof 7 *Assume $\lambda(p_1^{sub}) = \max\{\lambda(p_i^{sub})\}, i = 1...k$. Then,*
$a(p_i^{sub}) \leq \lambda(p_1^{sub}) \times w(p_i^{sub}), i = 1...k$. *Therefore,*
$\lambda(p) = \frac{\sum_i a(p_i^{sub})}{\sum_i w(p_i^{sub})} \leq \frac{\lambda(p_1^{sub}) \times \sum_i w(p_i^{sub})}{\sum_i w(p_i^{sub})} = \lambda(p_1^{sub}) = \max\{\lambda(p_i^{sub})\}$. *Similarly,* $\min\{\lambda(p_i^{sub})\} \leq \lambda(p)$.

This lemma allows bounding the density of a path using the information from its subpaths.

Definition 14 *An **active edge** is an edge $e \in E$ that has at least one activity. The end nodes of an active edge are **active node**.*

In Figure 3.3, active edges include (A, B), (C, H), (I, J), and (L, M), and active nodes are A, B, C, H, I, J, L, and M.

3.2.2 Problem Statement

The problem of Linear Hotspot Detection on All Simple Paths (LHDA) is formulated as follows:

Given:

1. A spatial network $G = (N, E)$ with a set of activities A, each associated with an edge,
2. A density threshold, θ_λ,
3. A p-value threshold, θ_{pv},
4. A number of Monte Carlo simulations m.

Find: Simple paths $p \in R$ where $\lambda(p) \geq \theta_\lambda$ and $pv(p) \leq \theta_{pv}$
Objective: Computational efficiency
Constraints:

1. Each $p \in R$ is longer than or equal to a minimum weight threshold θ_w,

2. Each $p \in R$ starts and ends with active nodes,

3. Results are correct and complete.

The input spatial network is defined in Definition 9. The inputs θ_λ and θ_{pv} are thresholds indicating the desired minimum density and level of statistical significance, respectively. Even though determining the optimal θ_λ is sometimes hard without domain knowledge, θ_{pv} can always be determined (e.g., 0.05 empirically) thanks to its statistical meaning. The input m is the number of Monte Carlo simulations for determining the p-value. Note that m depends on the precision of θ_{pv}. For example, a $\theta_{pv} = 0.01$ needs at least $m = 99$ simulations while a $\theta_{pv} = 0.001$ needs at least $m = 999$ simulations. Each simulation $MC_i, i = 1 \dots m$ randomly shuffles the activities on the spatial network and finds the maximum density of the simple paths using these random activities: λ_i^{MC}. Then, the statistical significance of a path p from the original activities is computed by ranking its density $\lambda(p)$ among $\lambda_i^{MC}, i = 1 \dots m$. More details are discussed in the baseline approach (Section 3.3).

The outputs for LHDA are all the simple paths between active nodes that meet both the density threshold (i.e. $\lambda \geq \theta_\lambda$) and the p-value threshold (i.e. $pv \geq \theta_{pv}$). The length of any linear hotspots is constrained to a value no less than a minimum weight threshold θ_w. This constraint eliminates meaningless tiny results such as a very short path with only one activity. Also, a hotspot must start and end with active nodes. This constraint eliminates meaningless results that have a dense subpath in the middle and empty subpaths hanging on the sides. Using the example in Figure 3.3 as input, and a density threshold $\theta_\lambda = 0.25$, a p-value threshold $\theta_s = 0.05$, and a minimum weight threshold $\theta_w = 5$, Path $p = \langle A, C, H, I, K, M, L \rangle$ is a linear hotspot with density $\lambda(p) = 0.28$, p-value $pv(p) = 0.03$, and weight $w(p) = 7.24$.

3.3 ASP_Base: baseline approach

In this section, we describe a baseline approach named ASP_Base to solve LHDA. It starts by enumerating and evaluating the density of all simple paths (ASPs) using depth-first-search (DFS) with backtracking over the input spatial network. After that, it sends the hotspot candidates (i.e. simple paths that survive both the density threshold θ_λ and

the minimum length threshold θ_w) to Monte Carlo simulations to find the statistically significant hotspots (i.e. p-value $\leq \theta_{pv}$).

Step 1: This step enumerates ASPs from the input spatial network. It takes each active node as the root and recursively traverses the network in a pre-order and depth-first manner. In each step of the traversal, a simple path p starting at the root node is enumerated and evaluated whether it is a hotspot candidate (i.e. $\lambda(p) \geq \theta_\lambda$ and $w(p) \geq \theta_w$).

Algorithm 6 shows the pseudocode of ASP_Base given a root node N. Inputs (1) to (4) are based on the problem statement. Input (5) is a stack structure that maintains the current path p_{cur} being enumerated, initialized as the root node. Input (6) is a hashmap which maps each node to a boolean value indicating whether a node is already in the current path p_{cur}.

First, ASP_Base traverses each neighbor node of the current node (Line 1). A neighbor node that is labeled "unvisited" N_{ne} now turns "visited" (Line 2) and is attached to the current path p_{cur} by pushing N_{ne} into p_{cur} (Line 3). If the new edge e connecting N and N_{ne} ($e = (N, N_{ne})$) is an active edge and the current path p_{cur} has both density and length larger than or equal to the input thresholds (Line 4), it is considered as a hotspot candidate and output in list C (Line 5). After that, ASP_Base recursively calls itself using N_{ne}, the updated p_{cur}, and $ifvisited$ as the inputs (Line 6). Once this neighbor node traversal is finished, the current path cannot be further extended. Thus, the current node N is labeled as "unvisited"(Line 7) and is removed from the current path stack by popping p_{cur} (Line 8).

By running this algorithm for every active node as the root, all the hotspot candidates in the input network G can be found.

Step 2: This step evaluates the statistical significance of each candidate hotspot using Monte Carlo simulations. First, m (e.g., 99) simulation datasets are generated, each randomly shuffling the input activities at new locations on the original spatial network G. Then, Step 1 is run on these simulation datasets and the highest density $d_i^{MC}, i = 1 \ldots m$ derived from each simulation is stored. The p-value indicating the statistical significance of a hotspot candidate is computed based on its rank among $d_i^{MC}, i = 1 \ldots m$. A hotspot candidate with a density smaller than l out of m values in d_i^{MC} has a p-value of $\frac{l+1}{m+1}$. Hotspot candidates whose p-values are less than or equal to

Algorithm 6 ASP_Base for a root node (recursive)

Input:
1) A spatial network G and activities
2) A root node N
3) A density threshold: θ_λ
4) A minimum weight threshold: θ_w
5) A stack storing current path: $p_{cur} \leftarrow N$
6) A boolean hashmap $ifvisited \leftarrow \{allnodes, false\}$

Output: C={Hotspots candidates starting at N}

Algorithm:

1: **For each** unvisited neighbor node of N: N_{ne} **do**
2: $ifvisited(N_{ne}) \leftarrow True$
3: $p_{cur} \leftarrow Push(N_{ne})$
4: **if** $\lambda(p_{cur}) \geq \theta_\lambda$ and $e = (N, N_{ne})$ is active and $w(p_{cur}) \geq \theta_w$ **then**
5: $C \leftarrow C \cup p_{cur}$
6: ASP_Base$(N_{ne}, \theta_\lambda, \theta_w, p_{cur}, ifvisited)$
7: $ifvisited(N) \leftarrow False$
8: $p_{cur} \leftarrow Pop()$
9: return

the given p-value threshold (e.g., 0.05) are deemed statistically significant hotspots.

3.4 ASP_FMGT: novel approach

ASP_Base solves LHDA using a depth-first-search (DFS) with backtracking. However, the cost is impermissible due to the huge number of simple paths being enumerated and evaluated. This section introduces a novel prune-and-refine algorithm based on bi-directional fragment-multi-graph traversal (ASP_FMGT) which substantially reduces the cost for solving LHDA without any loss of solution completeness and correctness.

3.4.1 Intuition behind ASP_FMGT

ASP_Base enumerates all simple paths (ASPs) from the given spatial network starting at active nodes. A large portion of these paths can be very sparse or even empty in real-world datasets. For example, the number of traffic accidents is much less than the number of road segments in a city. Based on this observation, we employ a pruning

phase which eliminates a large collection of sparse paths for a small computational cost, thereby substantially reducing the enumeration space of the following refine phase.

ASP_FMGT begins by partitioning the network into fragments and building an abstract graph where each node represents a fragment. Since there may be multiple edges connecting two fragments, we use a multi-graph to model the abstract graph, referred to as a fragment-multi-graph (FMG). Then, we build fragment-multi-graph trees (FMG-trees) which represent simple paths starting in the root FMG-node. After that, a bi-directional traversal is done on each FMG-tree to prune out sparse paths. Finally, in the refine phase, ASP_base is run on the pruned FMG-trees to find the statistical significant hotspots.

3.4.2 Building Fragment-multi-graph Trees

ASP_FMGT prunes the network by traversing an FMG-tree built based on the partitioned network.

Network partitioning: At first, the input spatial network is partitioned into two-dimensional rectangular grids. As partitioning is not the main focus of this chapter, other partitioning approaches are saved for future work. The network is partitioned evenly based on the coordinates of the input nodes and the number of fragments desired. For example in Figure 3.4 top left, the input network is partitioned into 2×2 fragments, namely w,x,y, and z. Since the nodes range from 1 to 5 on the x-axis and from 1 to 4 on the y-axis, the partitioning is at $x = 3$ and $y = 2.5$, respectively. Without any algorithmic effects, nodes at the partitioning border belong to the fragments in the lower index. For example, node $C = (3, 2)$ belongs to fragment w. Each fragment has boundary nodes which are neighbor of other fragments (e.g., node C in fragment w). The ASPs within each fragment are computed and the following necessary information is stored. **(1) maximum density from each boundary node to all other nodes ($\lambda_{ba}^{max}(N)$).** In fragment w, the maximum density from C to A and B is 0.41 and 0.31, respectively. Thus, this value of C is $\lambda_{within}^{max}(C) = \max\{0.41, 0.31\} = 0.41$. **(2) maximum density between each pair of boundary nodes ($\lambda_{bb}^{max}(N_1, N_2)$).** For example, in x, the maximum density between H and I is $\lambda_{bb}^{max}(H, I) = 0.83$.

Building Fragment-multi-graph: The original spatial network is now condensed

to an abstract fragment-multi-graph (FMG) where each FMG-node represents a fragment. Figure 3.4 top right shows the FMG of the example network. Each FMG-node (i.e. hexagon) is labeled with its boundary nodes and its associated fragment. An FMG has a much smaller size than the original network and thus allows traversal and pruning done with substantially less cost.

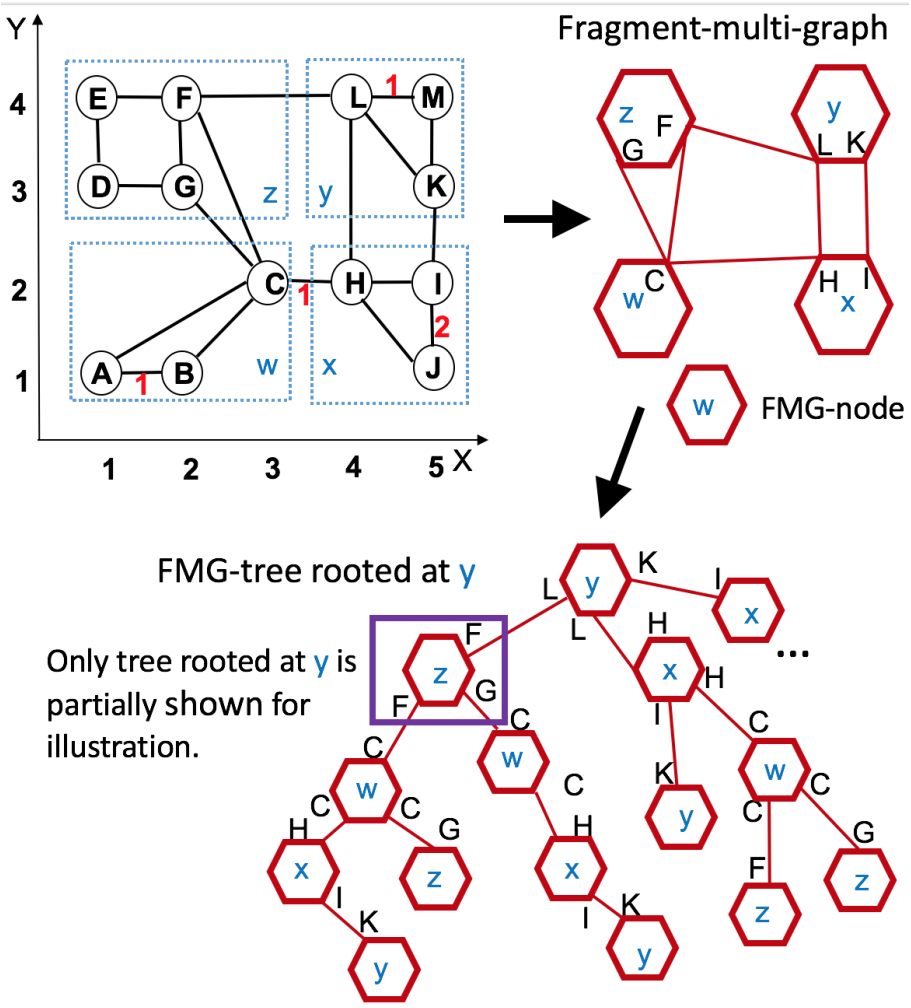


Figure 3.4: Example of network partitioning and fragment-multi-graph (best in color)

Building Fragment-multi-graph trees: Now that given the FMG, we build Fragment-multi-graph trees (FMG-trees) which cover ASPs in the original network. Similar to ASP_Base but applied on the FMG, an FMG-tree roots at an FMG-node

and recursively grows towards its neighbors. Figure 3.4 bottom shows a partial FMG-tree rooted at FMG-node y. For example, at start, y connects to fragments z and x via boundary node L as well as to fragment x via node K. These fragments connected from y are represented in the tree as three child FMG-nodes from the root. Note that the simple paths covered in the FMG-trees are a superset of the actual ASPs. An FMG-node can appear multiple times in a path in the FMG-tree since a simple path can pass a fragment multiple times even though each boundary node can only appear once. However, this ensures completeness of the solution with no adverse effects on correctness thanks to the refine phase at the end of the algorithm.

3.4.3 FMG-tree Pruning using bi-directional traversal

After building the FMG-trees which represent ASP, we now use a bi-directional traversal to prune each FMG-tree as well as its derived simple paths. We know from Section 3.4.2 that each FMG-node carries only the information about its corresponding fragment as introduced in the partitioning step (Section 3.4.2). However, in order to prune an FMG-subtree during traversal, we need to know more comprehensive information (e.g., upper-bound density) about the entire FMG-substree rooted at that node. We get this information by first using a bottom-up traversal that recursively gathers the information from the subtree rooted at each FMG-node. After that, a top-down traversal prunes the tree by computing a density upper-bound during each traversal step.

Bottom-up traversal for information gathering: The pseudocode of the recursive bottom-up traversal is shown in Algorithm 7. To start, each FMG-node considers the subpaths which pass through it and then end within one of its child FMG-nodes. For illustration, we use the subtree rooted at the FMG-node highlighted in purple rectangle in Figure 3.4. A path could reach this FMG-node z via F, then either directly exit z and enter w via C (left child) or pass through z, exit at G, and then enter w via C (right child). Each of these subpaths is concatenated by three segments: (1) the path from the inbound node to outbound node within the parent FMG-node, (2) the connecting edge between the parent and child FMG-nodes, and (3) the path within a child FMG-node from its inbound node. Therefore, according to Lemma 7, a "local" upper-bound density of this subpath λ^{max}_{local} is the maximum between the following three values: **(1) maximum density between the inbound node to all outbound nodes** λ^{max}_{within}

(Line 3). This value is the maximum value of λ_{bb}^{max} for the inbound node pre-computed during network partitioning. In the example, it is the largest density from inbound node F to G in fragment z: $\lambda_{bb}^{max}(F, G)$. **(2) maximum density of the connecting edges** λ_{edge}^{max} (Line 4). In the example, it is the larger density between $\lambda\,(e = (F, C))$ and $\lambda\,(e = (G, C))$. **(3) maximum density of the paths within a child FMG-node from the inbound node** λ_{child}^{max}(Line 5). This is equal to $\lambda_{ba}^{max}(N_{inbound})$ pre-computed in network partitioning. In the example, it is the maximum density from C to all the other nodes within fragment w. Note that if the investigated FMG-node is a leaf, then this value is 0 (Line 1-2).

Now each FMG-node has a "local" upper-bound density λ_{local}^{max} (Line 6). In order to know the maximum of these "local" densities among the subtree rooted at it $\lambda_{subtree}^{max}$, each FMG-node recursively collects the "local" upper-bounds from its children (Line 7). This "subtree" upper-bound density indicates the maximum possible density of a path going down the substree rooted at this FMG-node. The bottom-up traversal uses this upper-bound as well as the density of the paths already traversed to make a final decision on whether a subtree should be pruned.

Algorithm 7 Bottom-up traversal (recursive)

Input:
1) FMG-node N^{FMG}
2) FMG-tree T^{FMG}

1: **if** N^{FMG} is a leaf in T^{FMG} **then**
2: return 0
3: $\lambda_{within}^{max} \leftarrow \max\{\lambda_{bb}^{max}(N_{inbound}, N_{outbound})$ for $N^{FMG}\}$
4: $\lambda_{edge}^{max} \leftarrow \max\{\lambda_{connectingedge}\}$ for N^{FMG}
5: $\lambda_{child}^{max} \leftarrow \lambda_{ba}^{max}(N_{inbound})$ for N^{FMG}
6: $\lambda_{local}^{max} \leftarrow \max\{\lambda_{within}^{max}, \lambda_{edge}^{max}, \lambda_{child}^{max}\}$
7: $\lambda_{subtree}^{max} \leftarrow \max\{\lambda_{local}^{max}, \max_{N_{child}^{FMG}} \textit{bottom-up}(N_{child}^{FMG}, T^{FMG})\}$
8: return $\lambda_{subtree}^{max}$

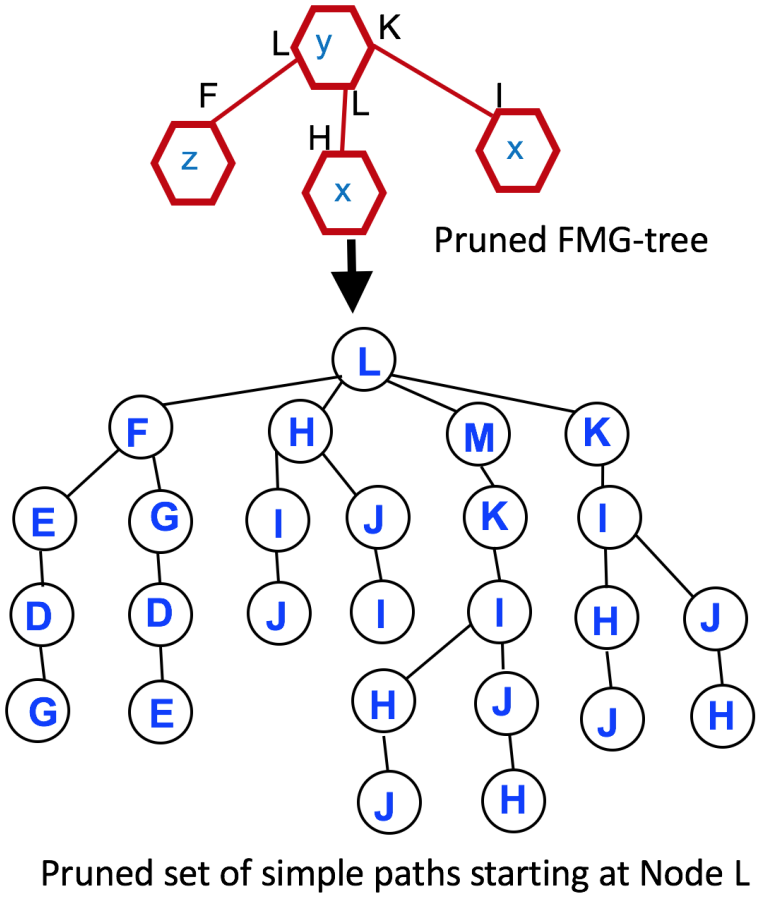


Figure 3.5: Example of pruned FMG-tree and the corresponding simple paths (best in color)

Top-down traversal for pruning: Following the bottom-up phase is the top-down traversal on the FMG-tree. At each stop at an FMG-node during the traversal, we decide whether we can prune the subtree rooted at this FMG-node or need to continue the traversal using the maximum value of the upper-bound densities of two parts: (1) already visited paths and (2) unvisited paths. For example, when the top-down traversal stops at the FMG-node highlighted by the purple rectangle ($FMG\text{-}node_{visited}$) shown in Figure 3.4, the upper-bound density is the maximum value of (1) any path from a node inside the root FMG-node to the inbound node of $FMG\text{-}node_{visited}$ (i.e. F) and (2) any path starting from $FMG\text{-}node_{visited}$. The first part is the concatenation of a

sequence of FMG-nodes and their connecting edges. Thus, its upper-bound density can be determined as the maximum density among the FMG-nodes and connecting edges in this visited path. The second part is computed during the bottom-up phase ($\lambda^{max}_{subtree}$). Now, we can prune the entire FMG-subtree rooted at an FMG-node being visited if the corresponding upper-bound density is smaller than the density threshold θ_λ. For example, given $\theta_\lambda = 1.5$, the FMG-tree in Figure 3.5 is pruned to be a much smaller tree with only four FMG-nodes as shown at the top of this figure.

3.4.4 Refine phase

The refine phase applies ASP_base on the pruned FMG-trees. It starts at each active node from the root FMG-node, and continues with the constraints of the pruned tree. Figure 3.5 bottom half shows the paths starting at L generated from the corresponding pruned FMG-tree. This tree is shown only for visualization and does not need to be stored.

Lastly, ASP_FMGT conducts Monte Carlo simulations for computing the statistical significance of each hotspot candidates.

3.5 Time and Time Complexity Analysis

We analyzed the space and time complexity for both ASP_Base and ASP_FMGT. To simplify the notations, we assume the numbers of nodes and edges are of the same magnitude. This assumption is valid for spatial networks since each road intersection usually connects less than 5 road segments. The notations in this analysis are: number of nodes (edges): $|N|$, number of active nodes: $f_{act} \times |N|$ where $0 \leq f_{act} \leq 1$, number of boundary nodes $|B|$, number of activities $|A|$, number of fragments $|F|$.

Time complexity Since Monte Carlo simulations multiply the entire cost by a constant number (e.g., 99), their cost is not considered in the analysis.

ASP_Base generates one simple path in each step during the DFS with backtracking. Therefore, its time complexity is equal to the number of simple paths in a network.

In the worst case when the network is a complete network, the cost is:

$$O\left(f_{act} \times \left(\frac{|N|!}{(|N|-2)!} + \frac{|N|!}{(|N|-3)!} + \ldots + \frac{|N|!}{0!}\right)\right)$$
$$= O\left(f_{act} \times |N|! \times \sum_{i=0}^{|N|-2} \frac{1}{i!}\right) = O\left(f_{act} \times |N|! \times e\right)$$

It can be seen that ASP_Base has factorial time complexity in the worst case. In most real cases, however, this number can be much smaller.

ASP_FMGT has costs from both the prune and refine phases. The cost of the prune phase includes the cost of partitioning the graph into rectangular fragments as well as building and depth-first-searching the FMG-trees. The partitioning includes assigning nodes to the fragments and computing ASPs in each fragment: $O\left(|N| + |F| \times \frac{|N|}{|F|}! \times e\right)$ assuming each partition has $\frac{|N|}{|F|}$ nodes. This cost will be dominated by the cost of the refine phase. The cost of building and searching the FMG-trees is equal to the size of the FMG-tree. In the case that the $|B|$ boundary nodes are evenly distributed in $|F|$ fragments, since each boundary node can only appear once on a path, the height of the tree is $\frac{|B|}{|F|}$. Thus, the cost is $O\left(e^{\frac{|B|}{|F|}}\right)$. The refine phase costs $O\left(f_{ref} \times f_{act} \times |N|! \times e\right)$ where f_{ref} is the percentage of simple paths that survive the prune phase ($0 \leq f_{ref} \leq 1$). Therefore, the total cost of ASP_FMGT is $O\left(e^{\frac{|B|}{|F|}} + f_{ref} \times f_{act} \times |N|! \times e\right)$ If nothing is pruned, ASP_FMGT costs more than ASP_Base does due to the cost from the zero-contribution prune phase. However, in practice, the cost saving is substantial because the vast majority of paths are pruned (e.g., $f_{ref} \approx 0.01$) as is demonstrated in the experiment analysis (Section 3.6).

Space complexity ASP_Base needs to store one simple path that is being investigated. This costs $O\left(|N|\right)$, which is the longest possible length of a simple path on the input network. In contrast, ASP_FMGT needs to store certain necessary information about each fragment plus one FMG-tree (Section 3.4.2). The dominant cost of storing information in each fragment is from storing the maximum density from each boundary node to each of the other nodes: $O\left(\frac{|B| \times |N|}{|F|^2}\right)$. Storing an FMG-tree costs an additional $O\left(e^{\frac{|B|}{|F|}}\right)$. Therefore, the entire space complexity of ASP_FMGT is $O\left(|F| \times \frac{|B| \times |N|}{|F|^2} + e^{\frac{|B|}{|F|}}\right)$. In summary, ASP_FMGT saves a substantial amount of

computational time with a relatively small amount of additional space compared to ASP_Base.

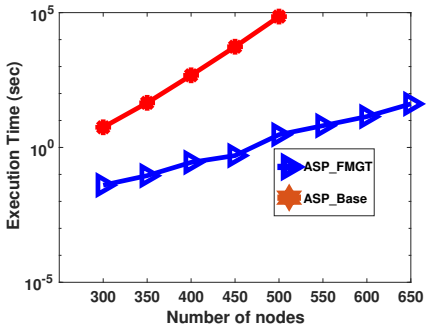
3.6 Experimental Evaluation

We conducted experimental analysis to evaluate the computational performance of ASP_FMGT and ASP_Base under a variety of parameters. The experiments aimed to answer the following questions: (1) What is the effect of **network size**? (2) What is the effect of **activity density**? (3) What is the effect of **number of partitions**? (4) What is the effect of **density threshold**?

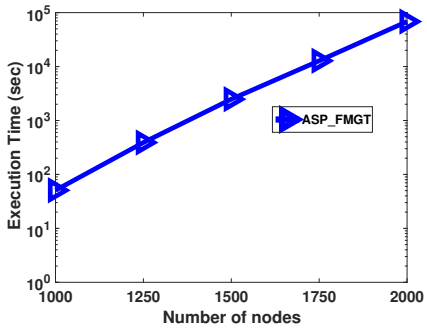
3.6.1 Experimental Setup

Each experiment varied one of the four parameters and set the rest to their default values. The spatial networks used in the experiments were synthetic networks whose nodes were distributed following complete spatial randomness and 95% of the nodes were 2-degree nodes, mimicking the real-world road networks It allows varying the size of the network easily without any sampling bias. The default number of nodes was 650. The activities were randomly distributed on the spatial network with a varied activity density (i.e. number of activities per edge). The default activity density was 0.01. The networks was partitioned in approximately equal size. The number of partitions was set to 50 by default. The default density threshold was set to 1. We did not include Monte Carlo simulation in the costs as they simply multiply the total cost by a constant (e.g., 99). The algorithms were implemented in Java and were executed on an Intel Core i7 2.5 GHz CPU and 16 GB RAM.

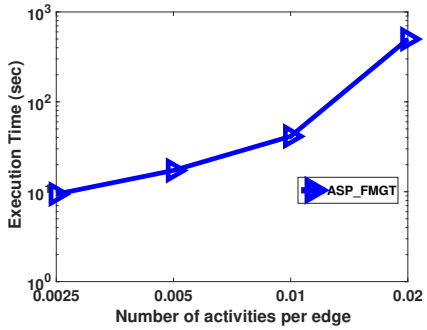
3.6.2 Experimental Results under Different Parameters



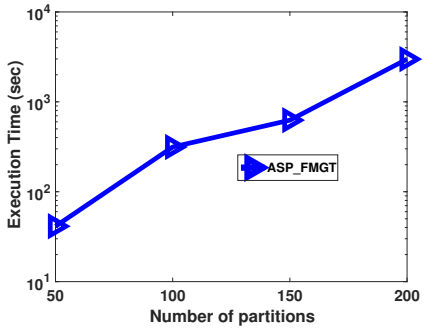
(a) Effect of network size (smaller)



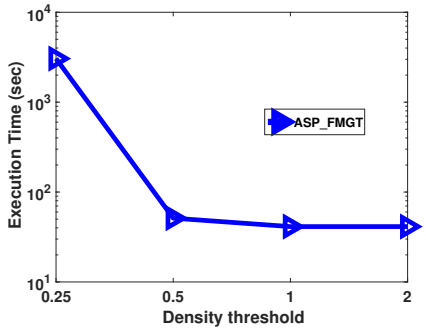
(b) Effect of network size (larger)



(c) Effect of activity density



(d) Effect of number of partitions



(e) Effect of density threshold

Figure 3.6: Experimental results under different parameters (y-axis in log scale)

Results are shown in Figure 3.6. The x-axis shows the parameters and the y-axis shows the execution times in log-scale.

Effect of network size (smaller networks): In this experiment, the performance of ASP_Base and ASP_FMGT by varying the network sizes while setting the other parameters to the default values are compared. Due to the expensive cost, ASP_Base was only tested on small networks with up to 500 nodes. As shown in Figure 3.6(a), ASP_FMGT was approximately 10^2 times faster than ASP_Base at start and the speedup got larger and larger and reached 10^5 faster with 500 nodes.

Effect of network size (larger networks): We also tested the performance of ASP_FMGT with larger networks ranging from 1000 to 2000 nodes while the other parameters use default values. Figure 3.6(b) shows that the increase of costs grew near linearly with the network size. In similar execution time (e.g, 10^5 seconds), ASP_FMGT handled a network 4 times larger than ASP_Base did.

More experiments were conducted to evaluate how the performance of ASP_FMGT is affected by different related parameters.

Effect of activity density: We evaluated ASP_FMGT with activity density varied from 0.0025 to 0.02 activity per edge. As shown in Figure 3.6(c), the cost first grew slowly and then faster as the density was larger. The reason was that the pruning rate remained very high at first and then dropped as the density increased further.

Effect of number of partitions: The number of partitions was varied from 50 to 200. As shown in Figure 3.6(d), the cost of ASP_FMGT first increased since the increased number of partitions led to larger FMG-trees and larger cost in the pruning phase.

Effect of density threshold: In this experiment, the density threshold was varied from 0.25 to 1. Figure 3.6(e) shows that the cost of ASP_FMGT dropped dramatically at first and then decreased slowly. This happened that the pruning rate significantly increased when the density threshold increased from 0.25 to 0.5. When the threshold increased further, the increase of pruning rate is small so the overall cost did not change a lot.

Overall, the experiments show that ASP_FMGT is much more efficient than ASP_Base. Also, the performance of ASP_FMGT under different varied factors were studied.

3.7 Case Studies on Real-world Datasets

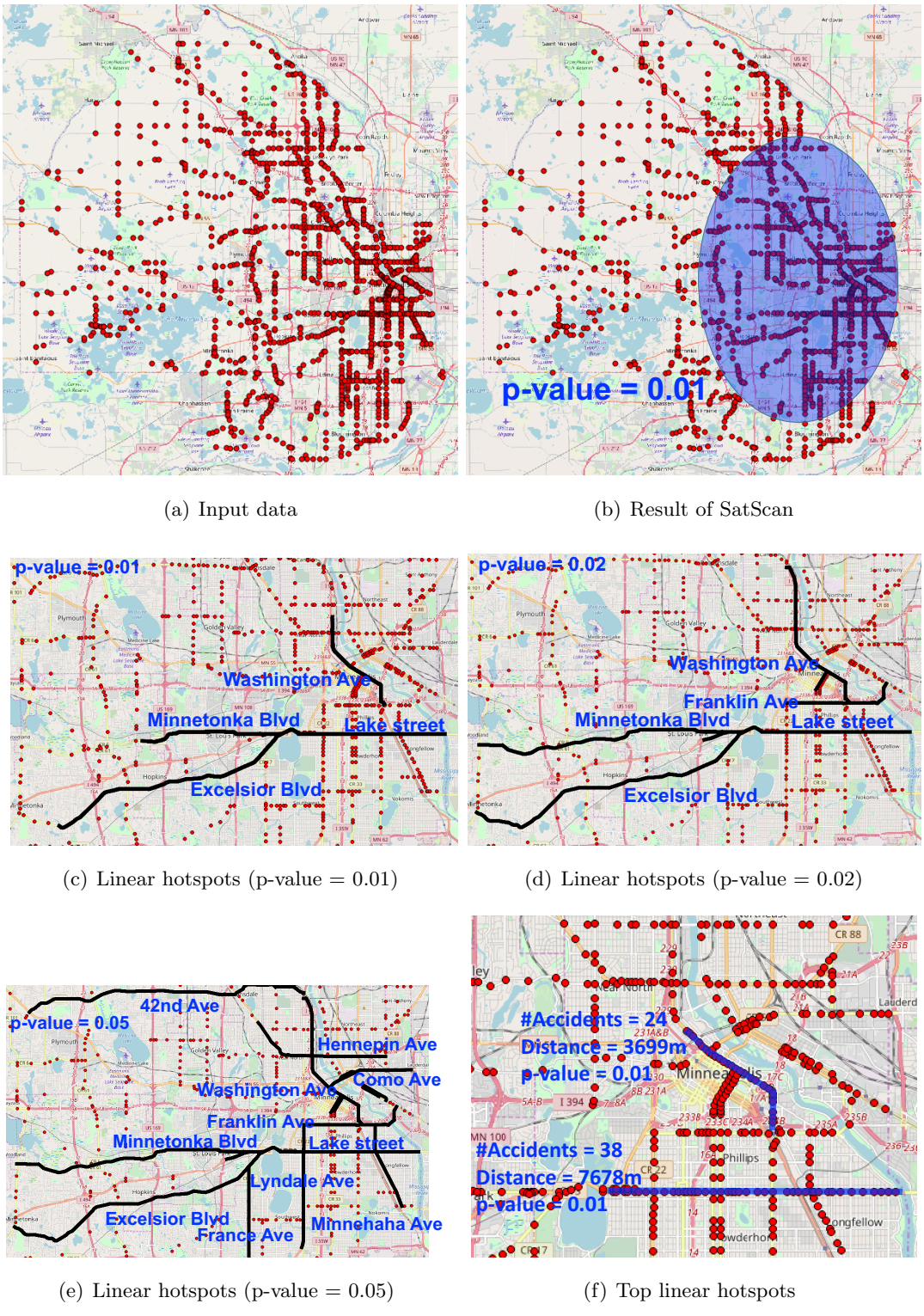
We conducted two case studies on real-world datasets [8, 3] to evaluate the effectiveness of the proposed ASP_FMGT. We compared ASP_FMGT to approaches including (1) SaTScan [69], the most popular spatial hotspot detection approach, (2) DBScan [89], a well studied clustering approach without statistical significance test, as well as (3) shortest path linear hotspot detection (LHDSP) [75]. The hotspots discovered were validated through groundtruth provided by studies in transportation safety [108].

3.7.1 Hennepin County Traffic Accident Dataset

In this case study, we evaluated the linear hotspots discovered by ASP_FMGT by comparing them to the result returned by SaTScan [69] and DBScan [89] on a real-world traffic accident dataset which contained 1345 traffic accidents occurred on major roads in Hennepin County (shown in Figure 3.7(a)), Minnesota, USA from 2010 to 2015 [8].

As shown in Figure 3.7(b), SatScan returns only one significant hotspot (p-value = 0.01) covering most of the metro area. This result does not provide useful information about individual roads prone to high traffic accident rates.

In contrast, our ASP_FMGT was able to locate the streets with statistically significant high accident density. Figures 3.7(c) to 3.7(e) are the maps of the linear hotspots using 0.01, 0.01, and 0.05 as the p-value threshold, respectively. For easier visualization, only results longer than 500 meters are shown. In Figure 3.7(c), four major roads, namely "Lake St", "Washington Ave", "Minnetonka Blvd", and "Excelsior Blvd" are the most significant hotspots (p-value = 0.01). As we increases the p-value threshold to 0.02 and 0.05, more hotspots are discovered. For instance, "Franklin Ave" and "Lyndale Ave" were discovered with p-value = 0.02 and 0.05, respectively. Moreover, Figure 3.7(f) shows the top two most significant hotspots, namely "Lake St" and "Washington Ave". These results can be validated according to previous studies published by the city of Minneapolis [108], which for example, identify "Lake St" and "Lyndale Ave" as the streets whose intersections have the highest pedestrian total number. Furthermore, we detected some new accident hotspots (e.g., Como Ave) that were previously unknown to domain experts. This represents valuable new information acquired at minimal cost, that domain experts can analyze to improve transportation safety.



(a) Input data

(b) Result of SatScan

(c) Linear hotspots (p-value = 0.01)

(d) Linear hotspots (p-value = 0.02)

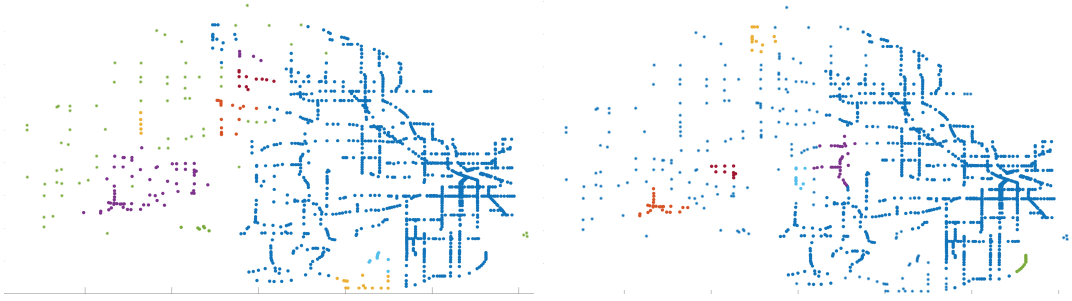
(e) Linear hotspots (p-value = 0.05)

(f) Top linear hotspots

Figure 3.7: Linear hotspots on traffic accidents in Hennepin County, Minneota [8]. (Best in color)

(a) DBScan results (eps = 0.01 degree, minPts = 5)

(b) DBScan results (eps = 0.01 degree, minPts = 10)



(c) DBScan results (eps = 0.02 degree, minPts = 5)

(d) DBScan results (eps = 0.02 degree, minPts = 10)

Figure 3.8: DBScan results on traffic accidents in Hennepin County, Minneota [8]. (Best in color)

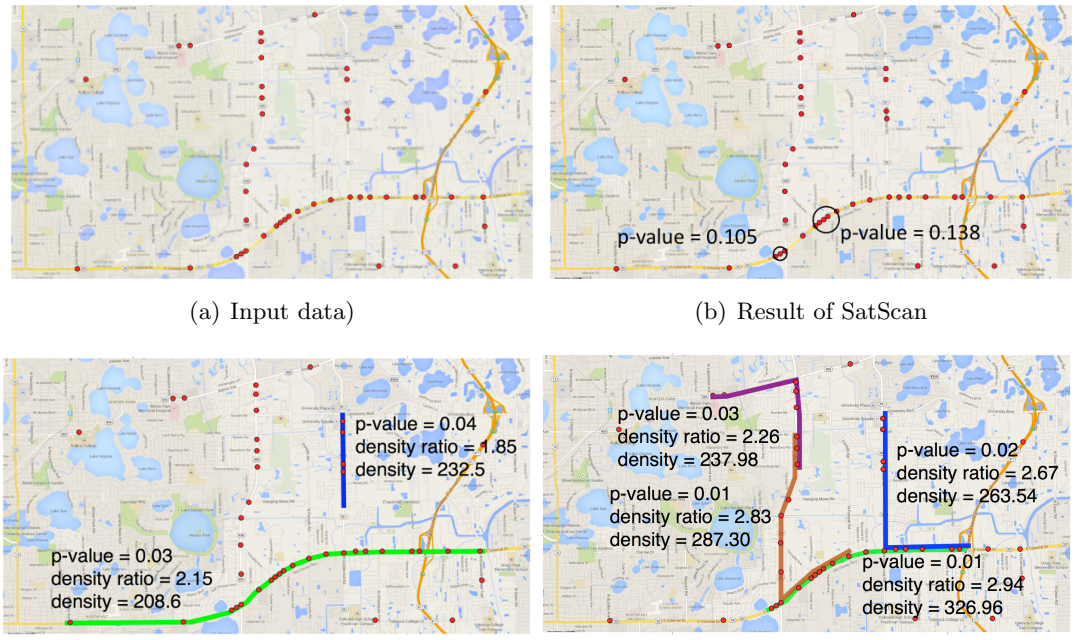
We also tested DBScan [89], which is traditionally used for identifying clusters without statistical significance test. We tuned DBScan using a range of parameters and highlighted the best results in Figures 3.8(a) to 3.8(d) using four combinations of parameters, namely eps = 0.01 and 0.02 degree and minPts = 5 and 10. Here we find that the results are very sensitive to parameters. In contrast to the statistically meaningful parameter (p-value) used in ASP_FMGT, it is difficult to determine the "correct" parameters. In addition, lack of a statistical significance test led to many false positives (e.g., those small clusters in Figure 3.8(a)) which cause users more effort to investigate and eliminate. Also, hotspots are represented as clusters with low accuracy. Small clusters usually covered only portions of hotspots and big clusters were usually too big

to make hotspots distinguishable. For example, Figure 3.8(b) shows a detected cluster (blue) covering the downtown area, but no specific streets such as "Lake St" was identified.

3.7.2 Orlando Pedestrian Fatality Dataset

We also compared ASP_FMGT with SaTScan [69] and Linear Hotspot Detection on Shortest Paths (LHDSP) [75] on a dataset of pedestrian fatalities occurred from 2000 to 2009 in Orlando, Florida [3]. Figure 3.9(a) shows 43 fatality cases (red dots) on the roads. Applied on this dataset, SatScan [69] returned two statistically insignificant circular patterns (p-values = 0.105 and 0.138) with a large empty area inside. SatScan [69] was not able to find any hotspots since it requires the hotspots in the shapes of circular or elliptical but the events occur on the road. With a p-value threshold of 0.05, LHDSP [75] found linear hotspots in shortest paths, as shown in Figure 3.9(c). However, when we compare these results to the hotspots found by ASP_FMGT (Figure 3.9(d)), it can be seen that LHDSP misses those patterns that are not shortest paths. Note that besides density, we also show density ratio [75] which is a popular concentration metric in network space for reference. It can be seen that ASP_FMGT found two new statistically significant hotspots (purple and orange) which are completely missed by LHDSP. Also, ASP_FMGT was able to discover those hotspots found by LHDSP with higher statistical significance and accuracy. The blue hotspot found by ASP_FMGT has a lower p-value since it contains the part on the bottom right which was missed by LDHSP. In addition, the green hotspot is now concentrated on a denser region with a lower p-value.

In summary, the case studies demonstrate that the ASP_FMGT outperformed popular spatial hotspot detection approach SatScan [69] and clustering approaches DBScan [89] by discovering linear hotspots validated by transportation engineering studies. Compared to the linear hotspots in shortest paths (LHDSP [75]), ASP_FMGT made improvements in both recall, by discovering new hotspots that were missing before, and precision, by increasing the accuracy of the hotspots found by LHDSP [75].



(a) Input data)

(b) Result of SatScan

(c) Shortest path linear hotspots

(d) All simple path linear hotspots

Figure 3.9: Case study on pedestrian fatalities in Orlando, Florida [3] (best in color)

3.8 Conclusion and Future Work

This chapter investigated the problem of linear hotspot detection on all simple paths (LHDA) which is important in many societal applications such as transportation engineering, public health, and public safety. In order to solve this computationally challenging problem, we proposed a novel approach ASP_FMGT based on bi-directional fragment-multi-graph tree traversal. Both theoretical and experimental analysis demonstrated that ASP_FMGT achieves substantial improvement in scalability compared to the baseline approach (ASP_Base) without any loss of completeness or correctness. Case studies on real-world datasets confirmed that not only does ASP_FMGT locate patterns with higher statistical significance and accuracy, it also finds statistically significant patterns completely missed by existing approaches.

In future, we plan to explore better network partitioning techniques that reduce the size of FMG-trees. We will start by investigating the min k-cut problem [109] since it leads to minimum boundary nodes and thus smaller FMG-trees. In addition, given the

#-p hardness of the LHDA problem, ASP_FMGT cannot deal with very large networks (e.g., million of nodes) in short execution time. Therefore, we plan to first develop parallel algorithms to scale up the solution. Our initial idea is to parallelize across independent Monte Carlo simulations and traversal on independent FMG-trees. We will also investigate further parallelization inside each FMG-tree for more performance improvement. Moreover, we plan to design scalable heuristics based on ASP_FMGT which minimally affect the solution quality . Specifically, we plan to impose paths in an FMG-tree to be without repeated fragments, reducing its maximum length by a factor of O(number of boundary nodes). Compared to ASP_FMGT, this heuristic will not enumerate the paths that pass a fragment back and forth. However, the most valuable hotspots without zigzag patterns (e.g., those shown in our case studies) will be retained. Finally, we plan to discover spatio-temporal linear hotspots which may potentially reveal the life-cycle and moving trends of hotspots using the additional temporal information.

Chapter 4

Spatio-temporal Intersection Detection from Trajectories with Data Gaps

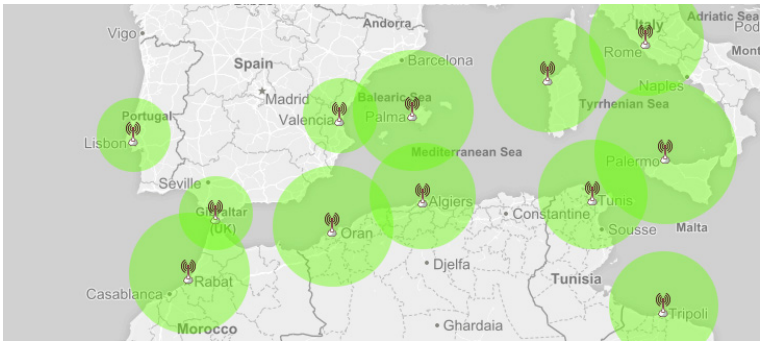
4.1 Introduction

Trajectory data generated from location tracking devices equipped on moving objects (e.g., ships, vehicles) have dramatically increased over the last decades. For instance, since the GPS transponders are equipped on ships more and more pervasively, the trajectories and other maritime attributes (e.g., drought) collected by Automatic Identification System (AIS) have been explosively increased [38]. Mining interesting patterns from these data has attracted significant attention from academia, industry, and government due to its important societal applications. This chapter studies the problem of Spatio-temporal Intersection Detection From Trajectories With Data Gaps (STID) which aims at identifying the novel Spatio-temporal Intersection (STI) patterns where the trajectories of the objects involved in the STI are missing. Distinct from the traditional patterns in trajectory mining, data are not available around the STI patterns.

4.1.1 Application Domains

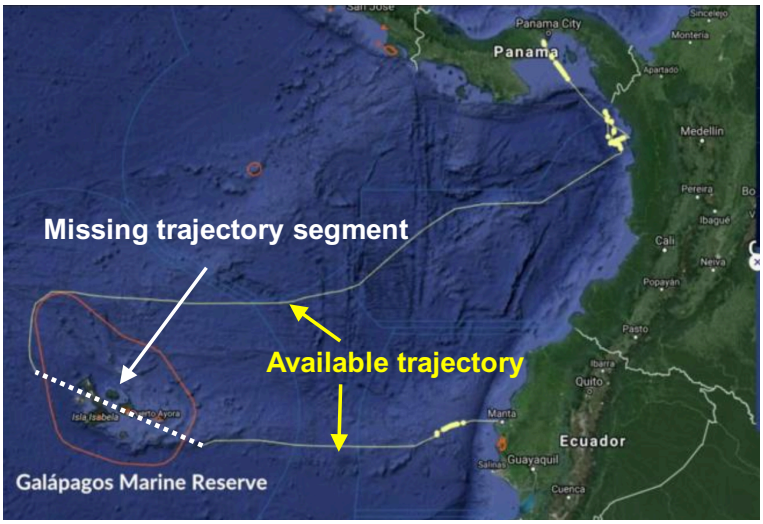
Ocean transportation is the most important method for global transportation over long distances [110]. Global container trade was approximately valued at 12 trillion U.S. dollars in the year of 2017 [110]. Also, the quantity of goods carried by containers has increased from around 100 million tons in 1980 to about 1.7 billion tons in 2015 [110]. Increasing maritime transportation posts critical needs of improving communication between ships and diminishing violation of regulations which affects safety and economy. STID helps address multiple problems in maritime safety and regulation using the trajectory data generated from Automatic Identification System (AIS) [38]. First, STID helps discover the regions of weak AIS signal coverage by locating the area containing frequent STI between missing trajectories. Figure 4.1(a) shows the terrestrial AIS coverage by FleetMon [9]. In addition, STID helps detect the maritime regulation violation where multiple ships stay together and intentionally hide their movements by switching their AIS transponders off [10]. For example, instead of directly loading ships from ports, two cargo ships may illicitly transfer goods on the ocean to avoid being documented. Figure 4.1(b) shows that a ship illegally transfers oil to another ship from North Korea, evading the United Nation sanction [10]. STID provides an efficient way to automatically identify such patterns from huge amount of AIS data which far exceed the capability of human eyes for monitoring and analyzing.

STID also helps detect illegal fishing which leads to billions of dollars of financial loss annually [111] by identifying the STI between the trajectories (with missing gaps) of the fishing vessels and the forbidden fishing zones. A real-world trajectory containing a missing segment shown in Figure 4.1(c) indicates a fishing vessel which switched off its tracking device before entering Galapagos Marine Reserve and then switched the device back on after conducting illicit fishing activities.



(a) AIS coverage map

(b) Illicit ship-to-ship transaction



(c) Illicit fishing with AIS device switched off

Figure 4.1: (a) Example AIS coverage map [9], (b) photo of an illicit ship-to-ship transaction [10], and (c) trajectory with missing segment near Galapagos Marine Reserve [11]

Besides maritime applications, STID can also be applied in improving homeland security and public safety on the ground. It is reported that leaders of terrorism organizations (e.g., AL-Qaeda) purposely shut off their communication devices when approaching to the meeting place [112], which is consistent with the definition of STI between the missing trajectories of leaders. Similarly, many other illegal activities such as underground drug deal can also be detected by STID.

4.1.2 Challenges

First, detecting STI is difficult since trajectory data are not available around the pattern, violating the assumption in traditional trajectory mining approaches [113, 114, 115, 116] that trajectory data are available around the pattern.

Moreover, the large amount of active objects and the generated trajectories make STID computationally challenging. For instance, MarineCadastre [38], one of the most popular well-structured maritime AIS datasets, records approximate $150,000$ ships around USA territory with a frequency of one GPS reading per minute from 2009 to 2014. The size of the entire dataset is about 600 GB. The computational cost discovering STI can be very expensive given this large amount of data. Assume there are $|N|$ objects, $|T|$ trajectories from each object, and each trajectory has $|M|$ points, the total number of GPS points is $|N| \times |T| \times |M|$. If $|E|$ consecutive missing GPS points are considered as an effective missing period, the number of events can be as many as $O(\frac{|T|\times|M|}{|E|})$ for each object. Therefore, the maximum number of STIs involving i objects is $\binom{|N|}{i} \times (\frac{|T|\times|M|}{|E|})^i$. Aggregating i from 2 to $|N|$, the total maximum possible number of STI is $\sum_{i=2}^{|N|} \binom{|N|}{i} \times (\frac{|T|\times|M|}{|E|})^i$.

4.1.3 Related Work

The most related research topic is Rendezvous Patterns Detection in the field of trajectory mining due to its important societal applications such as human behavior analysis and transportation planning [113]. Three rendezvous patterns are meeting [114], convergence [115], and encounter [116] which are defined based on the trajectory behavior of the objects involved [117]. A meeting pattern [114] consists of a set of objects moving towards a spatio-temporal cylinder and staying inside the cylinder for a period of time.

On top of that, a convergence pattern [115] enforces the consistency of the moving direction. As a special case of convergence, an encounter pattern [116] requires all the objects arrive at the same time. Existing approaches mining these patterns assume that the trajectories near the pattern are available and trusted. When a trajectory has gaps, they usually interpolate the data using the shortest path or learning from complete trajectories nearby [113]. However, these approaches are not designed to detect patterns when the trajectories of the moving objects are missing (e.g., due to weak signal or intentional movement hiding) in which case the objects possibly move far from the shortest path or the template trajectories. The proposed STID leverages the missing trajectory data to identify novel patterns that are not studied before.

Literature in the topic of time geography [118] models uncertainty between two trajectory sampling points as a bead or prism [119, 120, 121, 122, 123, 124, 125, 126, 127, 128], which brings uncertainty into some spatial database queries such as nearest neighbor [129, 130]. However, these studies cannot discover unknown informative patterns by interpreting the underlying incentives of the missing data and addressing the corresponding computational challenges.

4.1.4 Contribution

In this paper, we first formulate the problem of STID which discovers novel patterns from trajectories with missing segments based on concepts in time geography [119, 120, 121, 122, 123, 124, 125, 126]. Moreover, we propose a spatial join based STI detection algorithm (SJoin_STID) to solve STID. Then, we propose Apriori-graph traversal based algorithm (AGT_STID) which significantly improves the scalability while keeping the correctness and completeness of the solution. Results are qualitatively (i.e. via case study) and quantitatively (i.e. via experiments) evaluated using real-world data to demonstrate both the effectiveness and efficiency of the proposed algorithms.

4.1.5 Scope and Organization of This Chapter

The incentives of STIs output by the proposed approaches are to be interpreted by domain experts and the analysis from the domain perspective is beyond the scope. Also, this chapter does not consider the islands and land boundaries in the study area

which could affect results.

The rest of this chapter is organized as follows: Section 4.2 introduces the basic concepts and formulates the STID problem. Sections 4.3 and 4.4 propose SJoin_STID and AGT_STID, respectively. Section 4.5 provides theoretical proofs and asymptotic cost analysis. In Section 4.6, experimental evaluation and comparison were conducted in terms of scalability. A case study on real AIS data demonstrating the effectiveness of STID is discussed in Section 4.7. Section 4.8 concludes the chapter and previews the future work.

4.2 Problem Statement

This section starts by listing the necessary concepts including the bead model from time geography [119, 120, 123, 125, 126] with their definition and illustrative examples. Then, we formulate the problem of Spatio-temporal Intersection Detection From Trajectories With Data Gaps (STID).

4.2.1 Basic Concepts

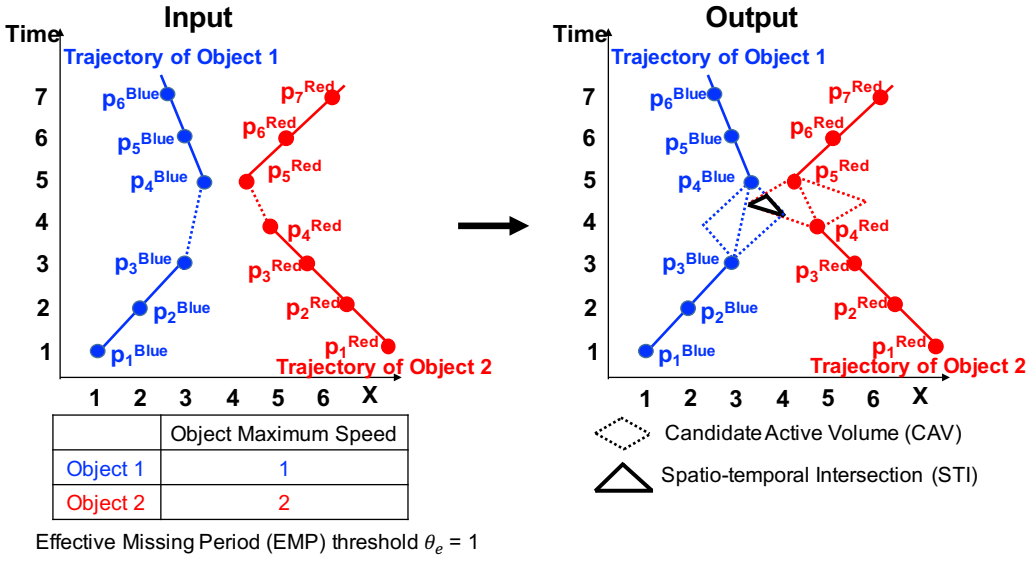


Figure 4.2: An illustrative example of STID (best in color)

Definition 15 *A **study area** is a two-dimensional rectangular area where the input data are located. It usually complies with the (latitude, longitude) coordinate system.*

In order for clear visualization using an example, without loss of generality, the study area is simplified to one-dimension space in the illustration in Figure 4.2 left.

Definition 16 *A **trajectory** is a chronological sequence of m spatio-temporal points $p_i = (x_i, y_i, t_i), i = 1 \dots m$ where each point is represented by its spatial coordinates (x_i, y_i) and a timestamp t_i.*

Figure 4.2 left shows the trajectories of Object 1 (blue) with 6 points (i.e. $p_i^{Blue}, i = 1 \dots 6$) and Object 2 (red) with 7 points (i.e. $p_i^{Red}, i = 1 \dots 7$).

Definition 17 *A **Object Maximum Speed** (S_{max}) is the maximum speed of an object based on the domain knowledge.*

For instance, for maritime data, S_{max} can be identified from publicly available information vessel databases [38]. For vehicles, humans, or animals, we can use the maximum physically allowed speed instead.

Definition 18 *An **Effective Missing Period (EMP)** is a time period when the signal is missing for longer than a user-specified EMP threshold θ_e.*

Assume $\theta_e = 1$ in Figure 4.2 right, there are an EMP of Object 1 between timestamps 3 and 5 and an EMP of Object 2 between timestamps 4 and 5. We use symbol EMP_i to represent an EMP with index i.

Definition 19 *A **Candidate Active Volume (CAV)** is the spatio-temporal volume where an object is possibly located during an EMP. A CAV must be derived from an EMP.*

The CAV derived an EMP is the intersection of two cones (i.e. a bead) whose vertices are the endpoints of the EMP and the radius $r(t) = S_{max} \times |t - t_{endpoint}|$ at time t are the product of the object maximum speed S_{max} and the time difference between t and the endpoints [130].

In Figure 4.2 right, Given $\theta_e = 1$, $S_{max}(Object1) = 1$, and $S_{max}(Object2) = 2$, the CAV shows as two parallelograms. When generalized to $2D$ study area, a CAV is the overlap between two cones whose vertices are the two endpoints of the EMP. Related details and algorithms are introduced in Section 4.3. We use symbol CAV_i to represent the CAV of EMP_i.

Definition 20 *A **Spatio-temporal Intersection (STI)** is represented by a time range tr and a set of CAVs: $STI = (tr, CAV_{i...j})$.*

For example, given a set of CAVs: $CAV_{1...n}$, each from a different object, $STI = (tr, CAV_{1...n})$ exists if each CAV_i overlaps with all the other CAVs $\in CAV_{1...n}$ within tr.

In Figure 4.2, a CR is represented as the black triangle which is the overlapped spatio-temporal volume between the two CAVs. We use symbol $STI_{i...j}$ to represent the STI derived from the intersection of $CAV_{i...j}$.

A STI can be intuitively considered as a spatio-temporal neighborhood "clique" where each object in the "clique" are spatio-temporal neighbor to all the other objects.

4.2.2 Problem Formulation

The problem of STID is formulated as:

Given:

1. A study area S,
2. A set of $|N|$ trajectories $T = t_1 \dots t_{|N|}$, each associated with an object,
3. An object Maximum Speed (S_{max}) for each object,
4. An effective Missing Duration Threshold (θ_e).

Find: All Spatio-temporal intersection (STI) patterns.

Objective: Computational efficiency.

Constraints: Correctness and completeness.

For example in Figure 4.2, given the study area (one-dimensional), two trajectories, two object maximum speeds, and $\theta_e = 1$, the output is the STI represented by the triangle shown in the figure.

4.3 SJoin_STID: Spatial Join Based Spatio-temporal Intersection Detection with Data Gaps

STID relates to computing the intersection between a set of Candidate Active Volumes (CAVs). One solution for STID is based on spatial join [131]. Traditional spatial join techniques mainly deal with very few (usually 2) data sources. However, these techniques are not directly applicable in the STID problem since it needs to join the trajectories from thousands of different ships. To overcome the limitation of existing spatial join techniques and solve the STID problem, We propose SJoin_STID which follows the framework of the plane sweep algorithm [132], but is able to compute the intersection between as many as objects needed.

4.3.1 Details of SJoin_STID

Algorithm 8 SJoin_STID

1: $ObservedObjectList\{\} \leftarrow \emptyset$
2: $Output\{\} \leftarrow \emptyset$
3: Sort the endpoints of all EMPs based on their x-axis coordinates
4: **For each** sorted endpoint p **do**
5: **if** p is the spatial start point of EMP_i **then**
6: $EMP_i \rightarrow ObservedObject\{\}$
7: Determine whether EMP_i intersects with each element $Object_j \in ObservedObject\{\}$
8: **if** intersected **then**
9: $STI = Intersection(EMP_i, Object_j)$
10: $STI \rightarrow ObservedObject\{\}$; $STI \rightarrow Output\{\}$;
11: **else**
12: Remove all the elements involving $EMP_i \in ObservedObject\{\}$

Algorithm 8 shows the pseudo-code of SJoin_STID.

Step 1: Sort the endpoints of all the Effective Missing Periods (EMPs) First, we sort the endpoints of all EMPs based on one of the coordinates. An endpoint is represented by three coordinates, namely x, y, and time, and either x or y can be the sorting coordinate. To be clear and consistent, we pick x throughout this chapter.

Step 2: A plane orthogonal to the x-axis sweeps along the sorted EMPs The second step conducts the sweeping. Imagine that there is a plane parallel to the y-t plane and orthogonal to the x-axis sweeping from the small end to the large end of the x-axis. The sweeping plane stops at both start and end endpoints of each EMP. Note that the "start" and "end" here mean on the order being swept which is irrelevant to the temporal dimension. An Observed Object List is maintained to store both the CAVs being currently crossed by the sweeping plane and all the STIs involving these EMPs. When stopping at the start of an EMP, it determines if a new STI is constructed from this EMP and each existed STI in the observed object list. The algorithm how to determine if a new STI can be constructed from an EMP and an existed STI is discussed in Section 4.3.2. If constructed, the new STI is added into the observed object as well as the outputs. On the other hand, when stopping at the end of an EMP, the algorithm removes all the STIs that involve the EMP from the list. To achieve constant time cost for adding and removing each object from the observed object list, we use a doubly-linked list to store the objects in the list as well as a hashmap (key = EMP, value = { all objects involving the key }) to quickly locate all the objects involving the key EMP in the list. Note that each of these STIs has different corresponded time periods indicating when the STI may happen. We introduce how to compute the time period in the following Section 4.3.2.

Line	Observed Object List
a (start of 1)	1
b (start of 2)	1, 2, <1, 2>
c (start of 3)	1, 2, <1, 2>, 3, <1, 3>, <2, 3>, <1, 2, 3>
d (start of 4)	1, 2, <1, 2>, 3, <1, 3>, <2, 3>, <1, 2, 3>, 4, <2, 4>
e (end of 1)	2, 3, <2, 3>, 4, <2, 4>
f (end of 2)	3, 4
g (end of 4)	3
h (end of 3)	Empty

Outputs: <1, 2>, <1, 3>, <2, 3>, <1, 2, 3>, <2, 4>
Note that the temporal information of the output STI is not shown

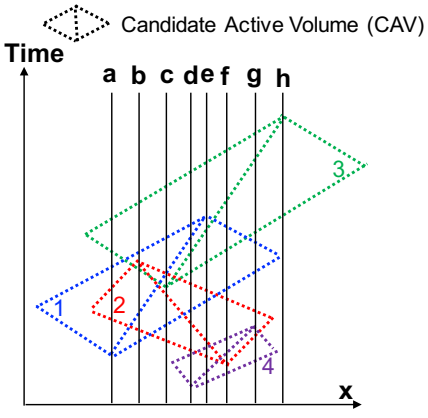


Figure 4.3: Execution example of SJoin_STID (best in color)

An execution trace of SJoin_STID:

Figure 4.3 shows a dataset contains EMPs and their corresponding CAVs from four objects. We simplify the study area into one-dimensional for illustrative purpose. A vertical line sweeps from left to right and stops at the endpoints (from a to h) of each EMP. The table on the right shows the elements in the observed object list after each stop. For example, when stopping at Line d (start of 4), the algorithm determines whether the incoming EMP<4> intersects with each element in the observed object list, namely <1>, <2>, <1, 2>, <3>, <1, 3>, <2, 3>, and <1, 2, 3>. Among those, since EMP <2> intersects with <4>, <4> and <2, 4> are added to the list. Also, <2, 4> is added to the output list as well. When stopping at Line e (end of 1), the algorithm removes all the elements in the list involving EMP 1 including <1>, <1, 2>, <1, 3>, and <1, 2, 3>. After that, the elements left are <2>, <3>, <2, 3>, <4>, and <2, 4>. The last stop is at Line h (end of 4). The Observed Object List becomes empty and the final output STIs include: <1, 2>, <1, 3>, <2, 3>, <1, 2, 3>, and <2, 4>.

4.3.2 Construct New Spatio-temporal Intersection (STI)

This subsection discusses the details of Line 7 in Algorithm 8 which determines whether an EMP intersects with either an EMP or a STI. A loop goes over all the EMPs involved in the input STI and check whether all of them intersect with the input EMP within the same time range. If they do, then a new STI is constructed with the new time range which is a subset of the time range of the input STI. Note that in each iteration, the intersection time range is updated to a subset of the previous range.

As defined in Section 4.3, an EMP is the intersection of two cones (i.e. a bead) vertexed at the endpoints of a CAV [130]. Therefore, in order to compute the intersection between two EMPs during each iteration of this loop, we need to determine the intersections between four cones (i.e. two beads). We first check if these two beads have overlapped time range. If not, then the two beads are guaranteed not intersected. Otherwise, the algorithm uses the following geometric property to determine the intersection. Figure 4.4 shows two beads generated from two EMPs. EMP_1 starts at $P_{start}^{EMP_1}$ and ends at $P_{end}^{EMP_1}$, while EMP_2 starts at $P_{start}^{EMP_2}$ and ends at $P_{end}^{EMP_2}$. Each point is represented by three coordinates. For example, $P_{start}^{EMP_1} = (x_{start}^{EMP_1}, y_{start}^{EMP_1}, t_{start}^{EMP_1})$

is presented by two spatial coordinates $x_{start}^{EMP_1}$ and $y_{start}^{EMP_1}$ as well as the temporal coordinate $t_{start}^{EMP_1}$. The radius of each cone at time t is the product of the object max speed S_{max} and the time difference from the start point of the EMP. For example, the radius of the cone vertexed at $P_{end}^{EMP_1}$ at time t: $r_{end}^{EMP_1} = (t_{end}^{EMP_1} - t) \times S_{max}^1$. Now, we formulate the sufficient and necessary condition that two bead intersect as follow:

$$r_{start} + r_{end} \leq dis(start, end) \tag{4.1}$$

where index $start = \{start^{EMP_1}, start^{EMP_2}\}$, index $end = \{end^{EMP_1}, end^{EMP_2}\}$, and $dis_{start,end}$ is the Euclidean distance between points P_{start} and P_{end}. Using known t and S_{max}, when $S_{max}^1 \geq S_{max}^2$, these equations are further derived into:

$$t \geq \frac{d_{s,e} + t_s \times S_{max}^1 + t_e \times S_{max}^2}{S_{max}^1 + S_{max}^2} \tag{4.2}$$

where $s = start^{EMP_1}$, $e = start^{EMP_2}$

$$t \geq \frac{d_{s,e} + t_s \times S_{max}^1 - t_e \times S_{max}^2}{S_{max}^1 - S_{max}^2} \tag{4.3}$$

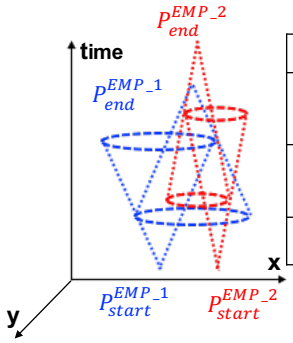
where $s = start^{EMP_1}$, $e = end^{EMP_2}$

$$t \leq \frac{d_{s,e} + t_s \times S_{max}^2 - t_e \times S_{max}^1}{S_{max}^2 - S_{max}^1} \tag{4.4}$$

where $s = end^{EMP_1}$, $e = start^{EMP_2}$

$$t \leq \frac{d_{s,e} + t_s \times S_{max}^1 - t_e \times S_{max}^2}{-S_{max}^1 - S_{max}^2} \tag{4.5}$$

where $s = end^{EMP_1}$, $e = end^{EMP_2}$. In the case that $S_{max}^1 < S_{max}^2$, the conditions are derived by swapping the variables. If the conditions are all satisfied, we know these two EMPs are intersected and the intersection time range the t bounded by the in-equations above. In contrast, if any of the conditions is not satisfied, the two EMPs are not intersected.



	Bead 1 (blue)	Bead 2 (red)
Start	$P_{start}^{EMP\_1} = (x_{start}^{EMP\_1}, y_{start}^{EMP\_1}, t_{start}^{EMP\_1})$	$P_{start}^{EMP\_2} = (x_{start}^{EMP\_2}, y_{start}^{EMP\_2}, t_{start}^{EMP\_2})$
End	$P_{end}^{EMP\_1} = (x_{end}^{EMP\_1}, y_{end}^{EMP\_1}, t_{end}^{EMP\_1})$	$P_{end}^{EMP\_2} = (x_{end}^{EMP\_2}, y_{end}^{EMP\_2}, t_{end}^{EMP\_2})$
Object max speed	S_{max}^{1}	S_{max}^{2}

Figure 4.4: Intersection of Two Effective Missing Periods (EMPs)

4.4 AGT_STID: Apriori-graph Traversal Based Spatio-temporal Intersection Detection with Data Gaps

In this section, we introduce a novel algorithm AGT_STID which detects STI substantially faster compared to SJoin_STID without any loss of solution completeness and correctness.

4.4.1 Intuition

There are redundancies in computing the intersection between the candidate active volume (CAV) of an input effective missing period (EMP) and all the objects in the observed object list in the SJoin_STID algorithm introduced in Section 4.3. Computing the later intersection repeats the computations for the earlier intersection. To eliminate such redundancy in computing the intersection between a set of STIs, we re-use the information stored when computing the intersection between the subsets of these STIs. Based on this idea, we propose a novel algorithm (AGT_STID) which models the observed objects as an Apriori-graph. This algorithm efficiently adds and deletes objects by traversing the Apriori-graph while re-using the information inherited from the predecessor nodes.

4.4.2 Details of AGT_STID

This sub-section discusses AGT_STID by the pseudo-code and execution trace.

Apriori-graph: Instead of a doubly-linked list, AGT_STID models the observed objects as a directed graph named Apriori-graph. Each node in the Apriori-graph stores an observed object along with an integer variable representing how many CAVs of this object are already computed intersected with a new input CAV (abbreviated as "count"). Note that this "count" considers the input CAV as well. Therefore, the count of a node is at least one. Once the count of an object reaches its size, this object is determined as a STI and is added to the final output along with its time range. The only exception is that any size-1 object is not a STI since each STI must be the intersection between CAVs from multiple objects.

Figure 4.5(a) shows the Apriori-graph of the example data in Figure 4.3 when the sweep line stops at Line c (start of 3). The root node serves as the entry of the graph which does not store an observed object. Three successor nodes of the root show store size-1 objects, namely CAV_1, CAV_2, and CAV_3, respectively. The "c" in the box of a node indicates its count. For example, the count of node CAV_1 is one. The size-2 nodes include $CAV_{1,2}$, $CAV_{2,3}$, and $CAV_{1,3}$, which are successor nodes of the size-1 nodes. All these size-2 nodes have a count of two which means that the two CAVs in the node are intersected. Similarly, the only size-3 object $CAV_{1,2,3}$ succeeds from the size-2 nodes and has a count of three.

There is an edge in the Apriori-graph if two conditions are satisfied. First, the start node represents a STI, in other words, the count of the node is equal to its size. Second, the objects in start node is subset of those in the end node. These conditions make it only need to check whether the input CAV intersects with a ST when the CAV is known intersected with the subsets of the STI. Note that this chapter uses the symbol $Node<i, \ldots, j>$ to represent an Apriori-graph node containing the intersection of $CAV_{i \ldots j}$. If $Node<i, \ldots, j>$ has a count equal to its size, it also represents the node of $CR_{i \ldots j}$. For example, $Node<1, 2>$ represents the node of $CR_{1,2}$ which is the intersection of CAV_1 and CAV_2.

AGT_STID retains the same framework as SJoin_STID which is based on a sweep line moving along one spatial dimension of the dataset. Next, we introduce the algorithms when the sweep line stops at the start or end of an EMP, respectively.

Adding Objects Involving an Input EMP: When the sweep line stops at the start of an EMP, we need to add all the nodes representing the STIs of the intersection between the input EMP and a node in the Apriori-graph.

Algorithm 9 shows the pseudo-code of the addition function. Figure 4.5(b) shows the Apriori-graph after adding objects involving EMP_4 as the sweep line stops at Line d (start of 4). First, a node with $c = 1$ of the input EMP_4 and an edge connecting from the root to this new node are created. After that, since $Node\langle 3\rangle$ is intersected with $Node\langle 4\rangle$, a new node representing $STI_{3,4}$ is created with $c = 2$. The edges connecting from the $Node\langle 3\rangle$ and $Node\langle 4\rangle$ to $Node\langle 3, 4\rangle$ are created. Then, a traversal from $Node\langle 3\rangle$ starts in the breadth-first manner. When a successor of $Node\langle 3\rangle$ has a count equal to its size, a new node of the successor and EMP_4 is created with $c = 2$ if not already existed. In contrast, if it is already existed, we add the count of the node by one. For example, since a successor, $Node\langle 2, 3\rangle$, has a count of 2 which is equal to its size, we create $Node\langle 2, 3, 4\rangle$ with $c = 2$. Another new node $Node\langle 1, 3, 4\rangle$ is also created with $c = 2$ since the successor $Node\langle 1, 3\rangle$ also has a count equal to its size.

Algorithm 9 Add nodes involving an input EMP in AGT_STID

Input:

1) CAV_{input}

2) The Root node $Node_{root}$ of an Apriori-graph

Output:

1) The Root of the updated Apriori-graph after moving objects involving EMP_{input}

2 The output set: $Output\{\}$

1: **For each** child node $Node^i_{child}$ of $Node_{root}$ **do**
2: Determine if CAV_{input} intersects with $Node^i_{child}$
3: **if** Intersected **then**
4: Create node of $Node^{size=2}_{new} = Intersection(CAV_{input}, Node^i_{child})$ with $count = 2$
5: $Node^{size=2}_{new} \rightarrow Output\{\}$
6: Create edge from CAV_{input} to $Node^{size=2}_{new}$
7: Create edge from $Node^i_{child}$ to $Node^{size=2}_{new}$
8: **For each** $Node_i$ in Breadth-first-search starting from $Node^i_{child}$ **do**
9: **if** $Node_i = Node^{size=2}_{new}$ **then**
10: Continue;
11: **if** $Node_i.count == Node_i.size$ **then**
12: **if** $Node_{new} = Intersection(CAV_{input}, Node_i)$ does not exist **then**
13: Create node of $Node_{new} = Intersection(CAV_{input}, Node_i)$ with $count = 2$
14: Create edge from $Node_i$ to $Node_{new}$
15: **if** $Node_i.size == 2$ **then**
16: Create edge from $Node^{size=2}_{new}$ to $Node_{new}$
17: **else**
18: $Node_{new}.count++$
19: **if** $Node_i.count == Node_i.size$ **then**
20: $Node_i \rightarrow Output\{\}$

One special thing we need to take care of is that when traversing the newly created

size-2 node (e.g., $Node<3,4>$), edges connect from it to its superset size-3 nodes without adding its count. In our example, edges connecting from $Node<3,4>$ to $Node<2,3,4>$ and $Node<1,3,4>$ nodes are created without adding the count of these two size-3 nodes. Next, the traversal continues to the successors of size-3 nodes. Only CAV_3 is intersected with the input CAV_4. It is not a STI since the count is not large enough. The algorithm terminates once the traversal starting from all the size-2 nodes is completed. During the traversal, STIs with count equal to size are also output.

Delete Nodes Involving an Input EMP: When the sweep line stops at the end of an EMP, we need to remove all the objects involving that EMP. Algorithm 10 shows the corresponding pseudo-code. First, from all the child nodes of the root node (i.e., size-1 objects), we find the one that stores the input EMP. Note that we use a hashmap to store the size-1 objects to make this step completed in constant time. Instead of storing all the objects in a hashmap in SJoin_STID, storing only size-1 objects substantially saves the space cost. After that, starting from the node just found, we remove the node and recursively operate the deletion on its child nodes. Figure 4.5(c) shows the Apriori-graph after deleting objects involving EMP_1 when the sweep line stops at Line e (end of 1).

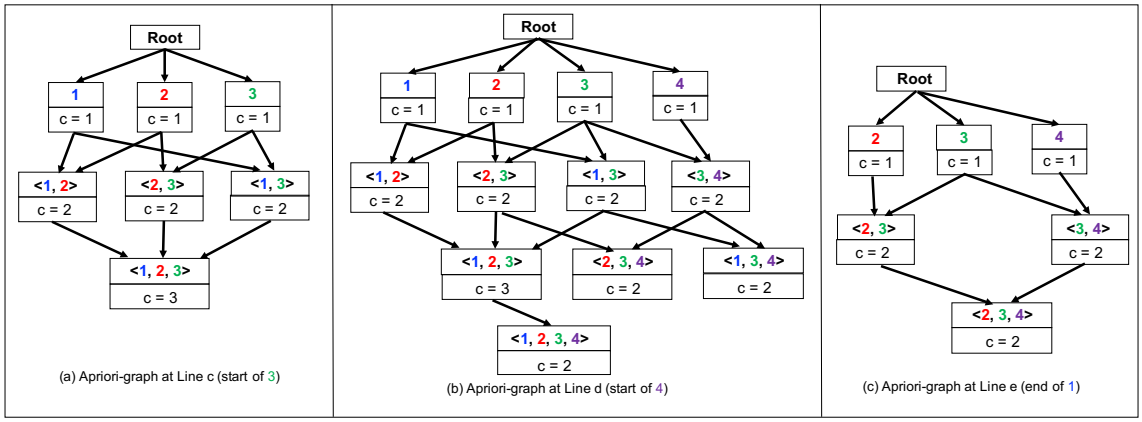


Figure 4.5: An illustrative example of AGT_STID

Algorithm 10 Delete nodes involving an input EMP in AGT_STID

Input:

1) EMP_{input}

2) The Root node $Node_{input}$ of an Apriori-graph

Output: The Root of the updated Apriori-graph after moving objects involving EMP_{input}

```
1: if Node_input is the root of the Apriori-graph then
2:     Retrieve the size-1 node Node_input that contains only EMP_input
3: else
4:     if Node_input has no child then
5:         delete Node_input
6:     else
7:         For each child node Node^i_child of Node_input do
8:             Recursively call this function with inputs EMP_input and Node^i_child
9:         delete Node_input
```

4.5 Theoretical Analysis

4.5.1 Correctness and Completeness of AGT_STID

Lemma 8 *Given a finite set of trajectories, AGT_STID terminates in finite time.*

Proof 8 *Given a finite set of $|N|$ trajectories, AGT_STID loops over a finite number of $2 \times |N|$ endpoints. Inside each iteration of the loop, AGT_STID conducts a depth-first-search on the Apriori-graph whose number of nodes are bounded by the finite number of STIs. Therefore, AGT_STID terminates in finite time.*

Lemma 9 *Given a clandestine rendezvous of n candidate active volumes: $CR_{old} = CAV_{i...n}$ and a new CAV: CAV_{new}, $CR_{new} = CAV_{i...n} \cup CAV_{new}$ is constructed if CAV_{new} intersects with all $CAV_{i...n}$ within a common time range that overlaps the time range of CR_{old}.*

Proof 9 *By definition, CR_{new} is constructed if all pairs of $CAV_{i...n}$ and CAV_{new} intersects within a common time range. All these pairs include two portions, namely pairs*

not involving CAV_{new} and pairs involving CAV_{new}. The pairs in the first portion are known intersected since the existence of CR_{old}. Also, the pairs in second portion are intersected each other with a common time range given the condition in this lemma. In addition, since the time ranges of these two portions overlap, CAV_{new} is constructed as a valid CR.

Lemma 10 *AGT_STID only outputs valid CR.*

Proof 10 *Now we prove that AGT_STID does not output any invalid STI which are not intersection between CAVs. (1) AGT_STID outputs an STI only when the corresponding node ($size > 1$) in the Apriori-graph has a full count (Line 18 in Algorithm 9). (2) A new node is created from an existed STI and an input CAV only when all the elements in the STI intersect each other (Line 12 in Algorithm 9). (3) A node has a full count only after we check all the elements in the STI intersect the input CAV (Line 18 in Algorithm 9). According to Lemma 9, the output STI is a valid CR.*

Lemma 11 *AGT_STID is correct.*

Proof 11 *Based on Lemmas 8, 9, and 10, AGT_STID outputs correct solution and terminates in finite time. Thus, AGT_STID is correct.*

Lemma 12 *AGT_STID does not underestimate the count for any node and does not miss any STI with a count equal to its size (i.e. full-count).*

Proof 12 *(1) AGT_STID creates edges to connect between each STI to all its subsets (Line 6-8, 14, 16 in Algorithm 9). Thus, the node involving an input EMP are guaranteed visited during the depth-first-search. (2) The count of a node is initialized as two (Lines 13 and 4 in Algorithm 9) and increases by one for each element intersected with the input EMP (Line 18 in Algorithm 9). Thus, the count representing the elements intersected each other is not underestimated. (3) AGT_STID outputs an STI once it has full-count (Lines 19-20 in Algorithm 9).*

Lemma 13 *AGT_STID does not delete any wrong CR.*

Proof 13 *When AGT_STID deletes all the nodes involving an input EMP (Lines 11-12 in Algorithm 8), it only traverses along the nodes containing super-sets of EMP (Lines 4-8 in Algorithm 10).*

Lemma 14 *AGT_STID is complete.*

Proof 14 *Based on Lemmas 12 and 13, AGT_STID outputs complete solution.*

4.5.2 Asymptotic Time Complexity of AGT_STID

In worst case, given k EMPs that are intersected within a spatio-temporal volume, the total number of objects (i.e. EMPs and STIs) in the observed object list is $\binom{k}{1} + \binom{k}{2} + \binom{k}{3} + \ldots + \binom{k}{k} = \sum_{i=1}^{k} \binom{k}{i}$. When a new $(k+1)$-th EMP comes, it needs to check whether intersected with all the existed k objects. Since checking with a size-i object costs i, the cost is $\sum_{i=1}^{k} i \times \binom{k}{i} = k \times 2^{k-1}$. Therefore, assuming there are $|N|$ EMPs in the dataset, the total cost for adding is $\sum_{k=1}^{|N|} k \times 2^{k-1} = O(N \times 2^{|N|})$. Now we analyze the complexity in better cases since the worst case rarely happens because of the sparsity of real datasets.

In best case where all the $|N|$ EMPs are temporally disjoint, the total cost is $O(|N|)$.

The realistic cases stay in between of the best case and worse case, where the number of EMPs simultaneously overlapped each other in average $|N_s| << |N|$. Thus, the total cost are bounded by $O(|N_s| \times 2^{|N_s|})$ in the case when all these $|N_s|$ CAVs are spatially overlapped. Therefore, the cost is much cheaper than exponential of the input size in practice.

We can observe that SJoin_STID and AGT_STID have the same worst case asymptotic time complexity bounded by the size of the output. However, in reality, AGT_STID usually achieves substantially higher efficiency compared to SJoin_STID. The computational saving comes from two aspects. First, SJoin_STID always need to check if the new EMP intersects with each object in the observed object list whereas AGT_STID only checks those whose parent nodes are valid CR. This significantly reduces the computational since invalid STIs tend to be in larger sizes. In addition, subsequent nodes directly inheriting the information from their parent nodes containing the subsets. For example, when computing the intersection between $CR{<}1 \ldots k{>}$ and $CAV{<}(k{+}1){>}$, AGT_STID can inherit the information from a parent node containing $CR{<}2 \ldots (k+1){>}$.

4.6 Experimental Analysis

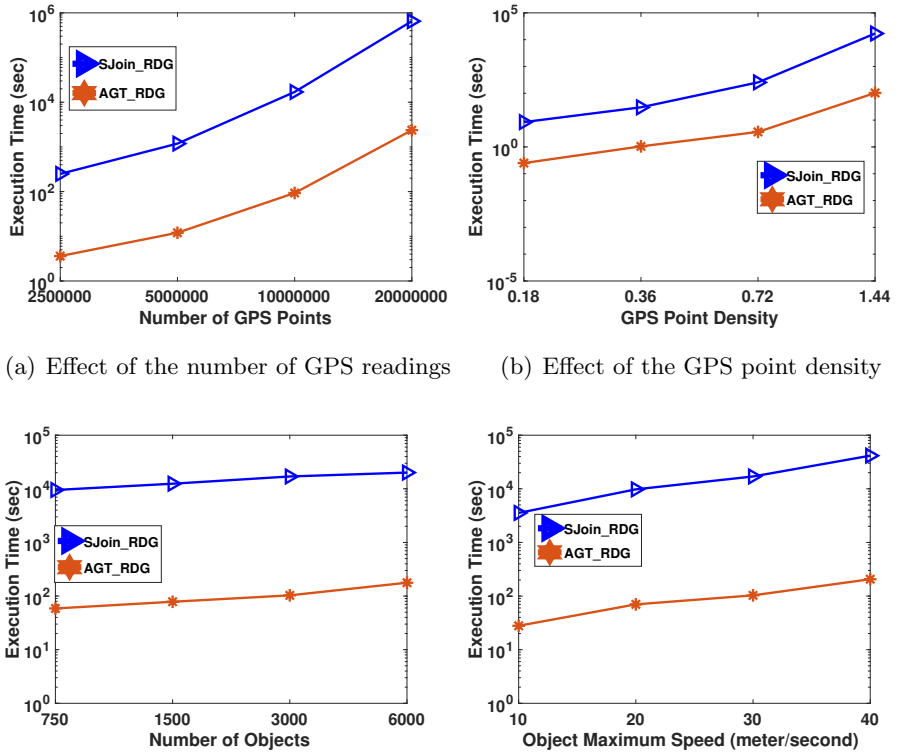
We evaluated the computational performance of SJoin_STID and AGT_STID under a variety of parameters using a real AIS dataset [38].

4.6.1 Experimental Setup

The **dataset** used in the experiments was MarineCadastre [38] which contained about 4×10^7 GPS readings from 6000 objects (i.e. ships). The **Experimental goal** was to evaluate the scalability of the proposed SJoin_STID and AGT_STID under different parameters. It was achieved by answering the following questions: (1) How does the number of GPS points affect the computational performance? (2) How does the GPS point density affect the computational performance? (3) How does the number of objects affect the computational performance? and (4) How does the object maximum speed affect the computational performance? The experiments were conducted by varying one of the following four parameters in each experiment: (1) number of GPS points (default $= 2 \times 10^7$), (2) GPS point density (default $= 1.44$ points$/KM^2$), (3) number of objects (default $= 3,000$), (4) object maximum speed (default 30 meters/second) while the others were set to default values. The costs shown in Figure 4.6 are in log-scale and the unit is second. All experiments were implemented in Java and performed on an Intel Core i7 2.5GHz CPU and 16GB memory.

4.6.2 Experimental Results

Effect of the number of GPS points: In this experiment, the numbers of GPS points were varied as 2.5×10^6, 5×10^6, 1×10^7, and 2×10^7. This was realized by picking a smaller study area from the original dataset for each varied number. Therefore, the density of GPS points remained the same as the original dataset. The results in Figure 4.6(a) show that the AGT_STID approximately is about 10^2 times faster Sjoin_STID and the advantage got larger when the number of GPS points increased.



(a) Effect of the number of GPS readings

(b) Effect of the GPS point density

(c) Effect of the number of objects

(d) Effect of the object maximum speed

Figure 4.6: Comparing execution times under different parameters

Effect of the GPS point density: In this experiment, the density of GPS points was varied as 0.18, 0.36, 0.72 and $1.44/KM^2$(GPS points are about 2.5×10^6, 5×10^6, 1×10^7, and 2×10^7) by randomly sampling the trajectories from the original dataset. The results in Figure 4.6(b) show that the gap between the costs of AGT_STID and Sjoin_STID gradually increased as the density increased. The reason is that increased density led to more STIs with larger number of objects. Thus, ACT_STID is more effective since it skip a lot of large STIs.

Effect of the number of objects: In this experiment, the number of objects (i.e. ships) were varied as 750, $1,500$, $3,000$ and $6,000$ (GPS points are about 2.5×10^6, 5×10^6, 1×10^7, and 2×10^7). The results in Figure 4.6(c) show that the speedup of AGT_STID compared to SJoin_STID approximately remained the same (i.e. 10^2 times faster).

Effect of object maximum speed (S_{max}): In this experiment, the object maximum speed were varied as $10m/s$, $20m/s$, $30m/s$ and $40m/s$ for all the objects. The results in Figure 4.6(d) show that the speedup provided by AGT_STID over Sjoin_STID slightly increased as S_{max} increased since larger S_{max} led to more and larger STIs which made AGT_STID provide more speedup.

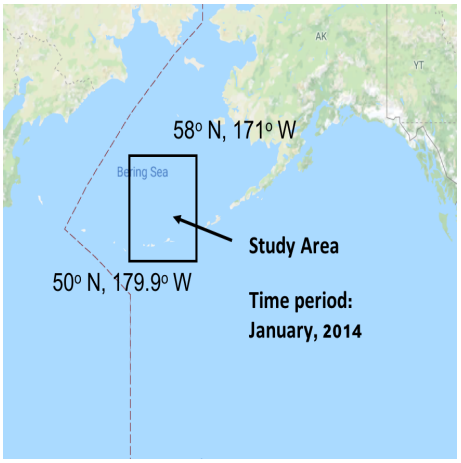
Overall, AGT_STID achieved substantial speedup (i.e. ˜10^2 times faster) over SJoin_STID and can handle ˜10^7 GPS points in ˜10^2 seconds.

4.7 Case Study on Real Automatic Identification System (AIS) Data

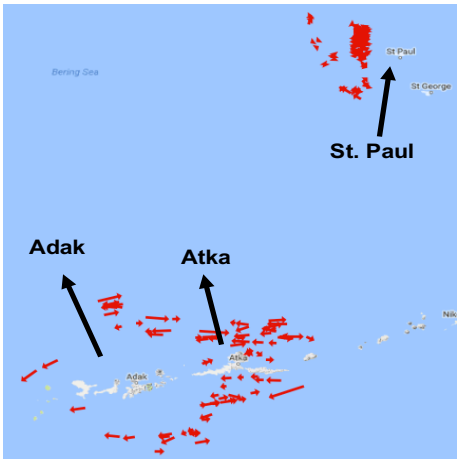
We conducted a case study on MarineCadastre, a popular real world AIS dataset [38]. The proposed approaches were applied on the study area ranging from $179.9W$ to $171W$ degrees in longitude and from $50N$ to $58N$ degrees in latitude in the Bering Sea, shown in Figure 4.7(a). It contained ˜1.4×10^6 GPS readings from 72 ships that traveled during January, 2014. The EMP threshold (θ_e) was set to 30 minutes by which only the top 0.5% longest missing periods were considered as EMPs. Figure 4.7(b) shows the EMPs which are represented by red arrows indicating the start and end points of the EMPs. Figure 4.7(c) shows the projected Candidate Active Volumes (CAVs) of the EMPs involved in a STI. Note that in order to show the 3D volume on the 2D study area, we plotted the 2D ellipses projected from the CAVs (in green color). The overlapped regions of the ellipses are the detected STI.

In Figure 4.7(c), we observed that there were two STI clusters. One was near the islands of "Adak" and "Atka" (bottom left), and another one located next to the island of "St. Paul" (top right). These patterns suggested weak and unstable AIS signal sent from the ships moving across the boundary of the effective zone of a terrestrial-AIS station (e.g., an island) since the AIS systems tended to switch between terrestrial-AIS and satellite-AIS [133] back and forth. Furthermore, we zoomed-in the results at the region below the island of "Adak" and identified two interesting patterns. On the left of Figure 4.7(d), there is large ellipse indicating a long signal missing period. However, the distance traveled during this period is short. Moreover, it is involved in a STI with another ship, possibly indicating a suspicious rendezvous that the participants

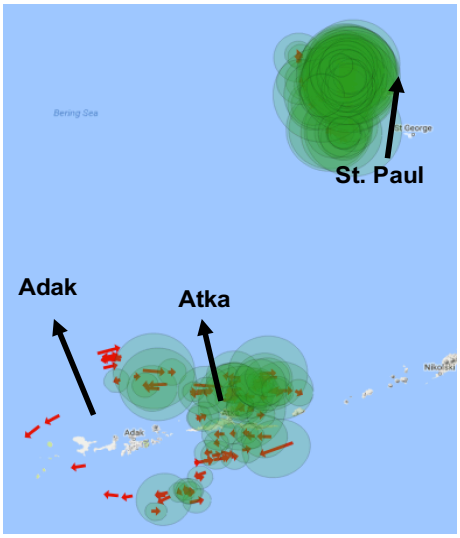
intentionally switched their AIS transponders off. On the right side, we can spot a region where a dense group of missing period located. This possible indicated a spot of weak AIS signal coverage. And, in the future, we plan to distinguish the patterns indicating weak AIS signal coverage and suspicious rendezvous.



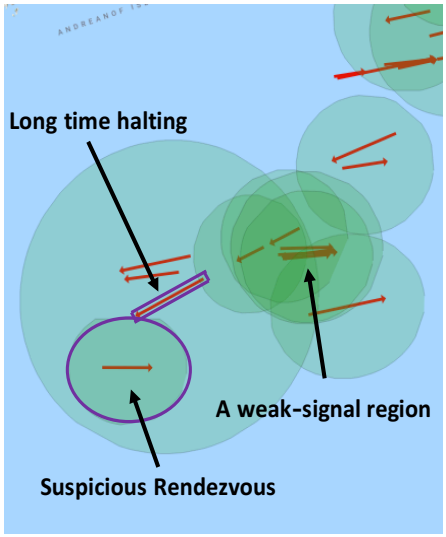
(a) Study area



(b) Effective missing segments



(c) Candidate active volumes and spatio-temporal intersections



(d) Zoom-in of the results

Figure 4.7: Spatio-temporal intersections detected in Bering Sea (best in color)

4.8 Conclusion and Future Work

We explored the Spatio-temporal Intersection Detection From Trajectories With Data Gaps (STID) problem which is critical in maritime safety and regulation, homeland security, and public safety, etc. The challenges of STID are two-fold. First, the trajectory data around the STI patterns are unavailable since GPS devices are intentionally switched off, violating the assumption in traditional trajectory mining approaches. Second, huge amount of trajectory data from of moving objects make STID a computationally expensive problem. To overcome the limitations of existing trajectory pattern mining approaches, we propose a spatial-join based algorithm (SJoin_STID) and an Apriori-graph traversal based algorithm (AGT_STID) which improves the computational performance without any loss of solution completeness and correctness. Experimental analysis shows that AGT_STID achieved substantial improvement in computational scalability over SJoin_STID and can handle ~10^7 GPS points. A case study on real AIS data demonstrated that the proposed approach was able to capture interesting patterns that are not found before.

Future work: In future, we plan to statistically measure the likelihood of patterns and conduct statistical significance test to eliminate chance patterns. In addition, we plan to evaluate our approach on more applications such as illegal fishing. Finally, besides a collection of STIs, we want to investigate "STI hotspots" spatio-temporal volumes where the frequency and density of STI is much higher inside the volume compared to outside. These hotspots may help improve the efficiency in deploying regulation enforcement forces.

Chapter 5

Conclusion and Future Directions

5.1 Key Results

The explosive growth of spatial data over the last decades emphasizes the need for data-driven approaches to discover spatial patterns for boosting many important domains such as mechanical engineering, transportation engineering, and public safety. As summarized in the taxonomy shown in Table 5.1, this thesis explored multiple data science techniques for mining patterns given various footprints of spatial events (e.g., spatial point events, spatio-temporal point events, and spatio-temporal linear events). Specifically, we investigated three novel and societally important patterns, namely linear hotspots on shortest paths, linear hotspots on all simple paths, and spatio-temporal intersections, all of which pose different statistical, computational, and mathematical challenges associated with them. For mining these spatial patterns, several novel algorithms were proposed which not only achieved computational efficiency but also followed statistical foundations and retained mathematical properties (e.g., correctness and completeness). Chapter 2 proposed a neighbor node filter and shortest tree pruning algorithm to prune unnecessary cost in enumerating shortest paths. Chapter 3 proposed a bi-directional fragment-multi-graph traversal for avoiding redundant computations in simple path enumeration. Chapter 4 proposed an apriori-graph structure

and its corresponding traversal algorithm to speed up the traditional plane sweep algorithm for detecting spatio-temporal intersection patterns. All the algorithms proposed were thoroughly evaluated and validated via both theoretical analysis (e.g., proofs and asymptotically computational cost analysis) and experiments using both real-world and synthetic datasets.

Table 5.1: A taxonomy of thesis contributions and future works

ST: Spatio-temporal		Pattern footprint				
		Point	Linear			Areal
			Shortest paths	Simple paths	All paths	
Event footprint	Spatial point event		Linear hotspots on shortest paths (**Chapter 2**)	Linear hotspots on all simple paths (**Chapter 3**) Algorithm parallelization **(Short term future work)**	Linear hotspots on trajectory-aware paths **(Short term future work)**	
	ST point event			Emerging hotspots on all simple paths **(Long term future work)**	Emerging hotspots on trajectory-aware paths **(Long term future work)**	
	ST linear event in trajectory			ST intersection summarization on all simple paths **(Long term future work)**	ST intersection summarization on trajectory-aware paths **(Long term future work)**	ST Intersections (**Chapter 4**) Statistically significant ST intersections **(Long term future work)** Areal ST intersection summarization **(Long term future work)**
	ST event in areal data					

5.2 Short Term Future Work

In the short term, as shown in Table 5.1, we plan to investigate (1) parallelization of algorithmss for mining linear hotspots on all simple paths and (2) linear hotspot discovery on trajectory-aware paths.

- **Algorithm parallelization for ASP_FMGT:** Given the #-p hardness of the problem of linear hotspot discovery on all simple paths, it is difficult to scale

up the current proposed ASP_FMGT to very large (e.g., country size) road networks. Therefore, we plan to develop parallel algorithms to significantly improve the scalability. First of all, thanks to the independence between Monte Carlo simulations, it is natural to distribute each Monte Carlo simulation to different computing units. In addition, the traversal of different FMG-trees in single run of ASP_FMGT do not rely on each other and can thus directly be parallelized. In the case of running 10^2 Monte Carlo simulations and partitioning the spatial network into 10^2 fragments, these two ideas could achieve a speed up of up to 10^4 times assuming the availability of enough computing units. we plan to explore more ideas for parallelizing other parts of ASP_FMGT such as inside each FMG-tree traversal for obtaining further speedup.

- **Linear hotspot discovery on trajectory-aware paths:** We plan to explore linear hotspots enumerated from all paths, which are even more complex than all simple paths. To avoid having search an infinite number of all paths due to the existence of cycles, we will look into the idea of using a finite number of paths collected from real-world trajectory data. These paths handle the situations that vehicles do not travel along shortest paths (e.g., bus routes) and, in the meantime, eliminate those simple paths that are actually not ever used. There is a lot of redundancy in the real-world huge trajectory data since many vehicles travel along similar roads and generate trajectories that share some portions. Therefore, we plan to summarize the trajectories by developing a statistically meaningful interest measure which indicates the popularity of paths so that the rarely used paths are eliminated from consideration In addition, we will investigate corresponding computational approaches which scale up the solution while keeping correctness and completeness.

5.3 Long Term Future Work

In the long term, as shown in Table 5.1, we will attempt to investigate more novel patterns including (1) emerging linear hotspots on all simple paths and trajectory-aware paths, (2) statistically significant spatio-temporal intersections, and (3) spatio-temporal intersection summarization for both areal and linear patterns.

- **Emerging linear hotspot discovery:** First, we plan to generalize and enrich the current linear hotspot patterns by incorporating temporal information. Specifically, we will investigate novel emerging hotspot patterns which model how hotspots form. This can be used for timely detection of hotspots such as disease outbreaks and traffic jams caused by accidents at their early stages. Detecting these spatio-temporal patterns is computationally more challenging compared to the spatial hotspot detection problems studied in this thesis due to the much larger enumeration space. we plan to investigate new algorithmic ideas to deal with emerging hotspots in trajectory-aware paths and all simple paths, respectively. Beside the formation of hotspots, the entire life cycle of hotspots will also be explored in the future.

- **Statistically significant spatio-temporal intersection discovery:** We plan to discover statistically significant spatio-temporal intersection patterns. Since the spatio-temporal intersection pattern studied in this thesis does not have a statistical significance test, it can generate false positive results and thus requires more examination by domain experts. To overcome this limitation, we will first develop an interest measure to represent the likelihood that a spatio-temporal intersection is indeed a pattern of interest such as an actual meet-up. This interest measure will consider multiple factors including duration of data mining, distance between involved objects and speed of objects. After the interest measure is developed, we will investigate new algorithms that address the computational challenges (e.g., caused by Monte Carlo simulations) for mining the novel statistically significant pattern.

- **Spatio-temporal intersection summarization:** Finally, we plan to explore patterns for summarizing individual spatio-temporal intersections such as their hotspots in both areal and linear footprints. In maritime applications, areal spatio-temporal intersection hotspots can assist in identifying regions where regulatory violations (e.g., illegal fishing, illicit transaction) frequently happen. Since spatio-temporal intersections are neither spatial nor spatio-temporal point events, and thus cannot be handled by existing hotspot detection approaches, novel interest measure and algorithms need to be investigated to address the challenges posed

by the new pattern. Besides areal hotspots, we will also study linear hotspots of spatio-temporal intersections in urban regions which can help point out routes where GPS tracking signals are frequently missing and possibly need improvement in infrastructure for communication (e.g., cellular tower). In addition to hotspots, other summarization patterns such as co-occurrence of spatio-temporal intersections will also be explored.

References

[1] Michelle Ernst, M Lang, and S Davis. Dangerous by design: solving the epidemic of preventable pedestrian deaths. *Transportation for America: Surface Transportation Policy Partnership, Washington, DC.*, 2011.

[2] Minnesota Department of Transportation. Bicycle and pedestrian safety initiative. `http://www.minnesotatzd.org/events/breakfasts/documents/bike_ped_initiatives.pdf`, 2015.

[3] Fatality analysis reporting system (fars) encyclopedia, national highway traffic safety administration (nhtsa). `ftp://ftp.nhtsa.dot.gov/fars/`, 2014.

[4] Martin Kulldorff. Spatial scan statistics: models, calculations, and applications. In *Scan statistics and applications*, pages 303–322. Springer, 1999.

[5] Lei Shi and Vandana P Janeja. Anomalous window discovery for linear intersecting paths. *Knowledge and Data Engineering, IEEE Transactions on*, 23(12):1857–1871, 2011.

[6] Marcelo Azevedo Costa, Renato Martins Assunção, and Martin Kulldorff. Constrained spanning tree algorithms for irregularly-shaped spatial clustering. *Computational Statistics & Data Analysis*, 56(6):1771–1783, 2012.

[7] Winnie Hu. No longer new york citys boulevard of death. `https://www.nytimes.com/2017/12/03/nyregion/queens-boulevard-of-death.html`, New York Times, 2017.

[8] Minnesota Department of Transportation. Minnesota crash mapping analysis tool. `https://www.dot.state.mn.us/stateaid/crashmapping.html`, 2018.

[9] Fleetmon live ais. https://blog.fleetmon.com/tag/fleetmon-live-ais/, 2018.

[10] Jake Kwon and James Griffiths. South korea seizes ship it claims transferred oil to north korea. https://www.cnn.com/2017/12/29/asia/north-korea-hong-kong-oil-intl/index.html, 2017.

[11] Jacqueline Savitz. How to spot the secretive activities of rogue fishing boats. http://www.bbc.com/future/story/20180607-how-to-spot-the-secretive-activities-of-rogue-fishing-boats, 2018.

[12] Shashi Shekhar, Steven K Feiner, and Walid G Aref. Spatial computing. *Commun. ACM*, 59(1):72–81, 2016.

[13] MaryAnne M Gobble. Big data: The next big thing in innovation. *Research-technology management*, 56(1):64–67, 2013.

[14] Li Zhou, Hao Wen, Radu Teodorescu, and David H.C. Du. Distributing deep neural networks with containerized partitions at the edge. In {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[15] Zonealloy: Elastic data and space management for hybrid SMR drives. In *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, Renton, WA, 2019. USENIX Association.

[16] Fenggang Wu, Baoquan Zhang, Zhichao Cao, Hao Wen, Bingzhe Li, Jim Diehl, Guohua Wang, and David HC Du. Data management design for interlaced magnetic recording. In *10th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, 2018.

[17] Zhichao Cao, Hao Wen, Fenggang Wu, and David HC Du. {ALACC}: Accelerating restore performance of data deduplication systems using adaptive look-ahead window assisted chunk caching. In *16th {USENIX} Conference on File and Storage Technologies ({FAST} 18)*, pages 309–324, 2018.

[18] ZHICHAO CAO, HAO WEN, XIONGZI GE, JINGWEI MA, JIM DIEHL, and DAVID HC DU. Tddfs: A tier-aware data deduplication-based file system tddfs: A tier-aware data deduplication-based file system.

[19] Zhichao Cao, Shiyong Liu, Fenggang Wu, Guohua Wang, Bingzhe Li, and David HC Du. Sliding look-back window assisted data chunk rewriting for improving deduplication restore performance. In *17th {USENIX} Conference on File and Storage Technologies ({FAST} 19)*, pages 129–142, 2019.

[20] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, volume 37, pages 164–177. ACM, 2003.

[21] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[22] Gaurav Makin, Kody Kantor, Hao Wen, Zhichao Cao, and Vallari Mehta. Systems and methods for performing live migrations of software containers, December 25 2018. US Patent App. 10/162,559.

[23] Hao Wen, David HC Du, Milan Shetti, Doug Voigt, and Shanshan Li. Guaranteed bang for the buck: Modeling vdi applications with guaranteed quality of service. In *2016 45th International Conference on Parallel Processing (ICPP)*, pages 426–431. IEEE, 2016.

[24] Hao Wen, David HC Du, Milan Shetti, Doug Voigt, and Shanshan Li. Guaranteed bang for the buck: Modeling vdi applications to identify storage requirements. *IEEE Transactions on Cloud Computing*, 2018.

[25] Hao Wen, Zhichao Cao, Yang Zhang, Xiang Cao, Ziqi Fan, Doug Voigt, and David Du. Joins: Meeting latency slo with integrated control for networked storage. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 194–200. IEEE, 2018.

[26] Bingzhe Li, Hao Wen, Farnaz Toussi, Clark Anderson, Bernard A King-Smith, David J Lilja, and David HC Du. Netstorage: A synchronized trace-driven replayer for network-storage system evaluation. *Performance Evaluation*, 130:86–100, 2019.

[27] Reem Y Ali, Venkata Gunturi, Shashi Shekhar, Ahmed Eldawy, Mohamed F Mokbel, Andrew J Kotz, and William F Northrop. Future connected vehicles: challenges and opportunities for spatio-temporal computing. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 14. ACM, 2015.

[28] N. Levine. Crime mapping and the Crimestat program. *Geographical Analysis*, 38(1):41–56, 2006.

[29] Reem Y Ali, Venkata MV Gunturi, Andrew J Kotz, Shashi Shekhar, and William F Northrop. Discovering non-compliant window co-occurrence patterns: A summary of results. In *International Symposium on Spatial and Temporal Databases*, pages 391–410. Springer, 2015.

[30] Reem Y Ali, Venkata MV Gunturi, Andrew J Kotz, Emre Eftelioglu, Shashi Shekhar, and William F Northrop. Discovering non-compliant window co-occurrence patterns. *GeoInformatica*, 21(4):829–866, 2017.

[31] P. Stolorz, H. Nakamura, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.Y. Chien, R. Mechoso, and J.D. Farrara. Fast Spatio-Temporal Data Mining of Large Geophysical Datasets. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, AAAI Press, 300-305*, 1995.

[32] Shashi Shekhar and Sanjay Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.

[33] S. Shekhar, M.R. Evans, J.M. Kang, and P. Mohan. Identifying patterns in spatial information: A survey of methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):193–214, 2011.

[34] Yiqun Xie, Emre Eftelioglu, Reem Ali, Xun Tang, Yan Li, Ruhi Doshi, and Shashi Shekhar. Transdisciplinary foundations of geospatial data science. *ISPRS International Journal of Geo-Information*, 6(12):395, 2017.

[35] Shashi Shekhar, Zhe Jiang, Reem Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[36] NASA. Sam: Smart assistant for earth science data mining. `https://earthdata.nasa.gov/community/community-data-system-programs/access-projects/sam`, 2018.

[37] Martin Kulldorff, Farzad Mostashari, Luiz Duczmal, W Katherine Yih, Ken Kleinman, and Richard Platt. Multivariate scan statistics for disease surveillance. *Statistics in Medicine*, 26(8):1824–1833, 2007.

[38] MarineCadastre.gov. Vessel traffic data. `https://marinecadastre.gov/ais/`.

[39] Gary Marcus and Ernest Davis. Eight (no, nine!) problems with big data. *The New York Times*, 6(04):2014, 2014.

[40] Peter M Caldwell, Christopher S Bretherton, Mark D Zelinka, Stephen A Klein, Benjamin D Santer, and Benjamin M Sanderson. Statistical significance of climate sensitivity predictors obtained by data mining. *Geophysical Research Letters*, 41(5):1803–1808, 2014.

[41] Sudipto Banerjee, Bradley P. Carlin, and Alan E. Gelfrand. *Hierarchical Modeling and Analysis for Spatial Data*. CRC Press, 2003.

[42] O. Schabenberger and C.A. Gotway. *Statistical Methods for Spatial Data Analysis*. Chapman and Hall, 2005.

[43] Alan E Gelfand, Peter Diggle, Peter Guttorp, and Montserrat Fuentes. *Handbook of spatial statistics*. CRC Press, 2010.

[44] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1993.

[45] W.R. Tobler. *Cellular Geography, Philosophy in Geography.* Gale and Olsson, Eds., Dordrecht, Reidel, 1979.

[46] Brian D Ripley. Modelling spatial patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 172–212, 1977.

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

[48] AS Fotheringham, C. Brunsdon, and M. Charlton. *Geographically weighted regression: the analysis of spatially varying relationships.* Wiley, 2002.

[49] C. E. Warrender and M. F. Augusteijn. Fusion of image classifications using Bayesian techniques with Markov rand fields. *International Journal of Remote Sensing*, 20(10):1987–2002, 1999.

[50] L. Anselin. Local indicators of spatial association-lisa. *Geographical Analysis*, 27(2):93–155, 1995.

[51] Stan Openshaw. *The modifiable areal unit problem.* ISBN 0860941345. OCLC, 1983.

[52] S. Shekhar, C.T. Lu, and P. Zhang. A unified approach to detecting spatial outliers. *GeoInformatica*, 7(2), 2003.

[53] Charu C Aggarwal. *Outlier analysis.* Springer Science & Business Media, 2013.

[54] J. Haslett, R. Bradley, P. Craig, A. Unwin, and G. Wills. Dynamic graphics for exploring spatial data with application to locating global and local anomalies. *American Statistician*, pages 234–242, 1991.

[55] Xutong Liu, Feng Chen, and Chang-Tien Lu. On detecting spatial categorical outliers. *GeoInformatica*, 18(3):501–536, 2014.

[56] Xutong Liu, Feng Chen, and Chang-Tien Lu. Spatial categorical outlier detection: pair correlation function based approach. In *19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011, November 1-4, 2011, Chicago, IL, USA, Proceedings*, pages 465–468, 2011.

[57] Dechang Chen, Chang-Tien Lu, Yufeng Kou, and Feng Chen. On detecting spatial outliers. *GeoInformatica*, 12(4):455–475, 2008.

[58] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Detecting spatial outliers with multiple attributes. In *15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), 3-5 November 2003, Sacramento, California, USA*, pages 122–128, 2003.

[59] Yufeng Kou, Chang-Tien Lu, and Dechang Chen. Spatial weighted outlier detection. In *SDM*, pages 614–618, 2006.

[60] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Min. Knowl. Discov.*, 28(1):190–237, 2014.

[61] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering co-location patterns from spatial datasets: A general approach. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(12):1472–1485, 2004.

[62] Y. Chou. *Exploring Spatial Analysis in Geographic Information System*. Onward Press, 1997.

[63] Zhi Liu and Yan Huang. Mining co-locations under uncertainty. In *Advances in Spatial and Temporal Databases - 13th International Symposium, SSTD 2013, Munich, Germany, August 21-23, 2013. Proceedings*, pages 429–446, 2013.

[64] Yan Huang, Jian Pei, and Hui Xiong. Mining co-location patterns with rare events from spatial data sets. *GeoInformatica*, 10(3):239–260, 2006.

[65] Song Wang, Yan Huang, and Xiaoyang Sean Wang. Regional co-locations of arbitrary shapes. In *SSTD*, pages 19–37, 2013.

[66] Wei Ding, Christoph F. Eick, Xiaojing Yuan, Jing Wang, and Jean-Philippe Nicot. A framework for regional association rule mining and scoping in spatial datasets. *GeoInformatica*, 15(1):1–28, 2011.

[67] Yan Li and Shashi Shekhar. Local co-location pattern detection: A summary of results. In *10th International Conference on Geographic Information Science (GIScience 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[68] Sajib Barua and Jörg Sander. Mining statistically significant co-location and segregation patterns. *IEEE Trans. Knowl. Data Eng.*, 26(5):1185–1199, 2014.

[69] Martin Kulldorff, Katherine Rand, Greg Gherman, Gray Williams, and David DeFrancesco. SaTScan v 2.1: Software for the spatial and space-time scan statistics. *Bethesda, MD: National Cancer Institute*, 1998.

[70] Daniel B Neill and Andrew W Moore. A fast multi-resolution method for detection of significant spatial disease clusters. In *Advances in Neural Information Processing Systems*, page None, 2003.

[71] Ganapati P Patil and Charles Taillie. Upper level set scan statistic for detecting arbitrarily shaped hotspots. *Environmental and Ecological statistics*, 11(2):183–197, 2004.

[72] Toshiro Tango, Kunihiko Takahashi, and Kazuaki Kohriyama. A space–time scan statistic for detecting emerging outbreaks. *Biometrics*, 67(1):106–115, 2011.

[73] Emre Eftelioglu, Shashi Shekhar, James M Kang, and Christopher C Farah. Ring-shaped hotspot detection. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3367–3381, 2016.

[74] Xun Tang, Emre Eftelioglu, and Shashi Shekhar. Elliptical hotspot detection: a summary of results. In *Proceedings of the 4th International ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, pages 15–24. ACM, 2015.

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

[76] 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. Springer, 2017.

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

[78] Jerry Ratcliffe. Crime mapping: spatial and temporal challenges. In *Handbook of quantitative criminology*, pages 5–24. Springer, 2010.

[79] Stan Openshaw, Martin Charlton, Colin Wymer, and Alan Craft. A mark 1 geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information System*, 1(4):335–358, 1987.

[80] S. Shekhar, P. Zhang, Y. Huang, and R.R. Vatsavai. Trends in spatial data mining. *Data mining: Next generation challenges and future directions*, 2003.

[81] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: Concepts, methodologies, and applications. *ACM TIST*, 5(3):38:1–38:55, 2014.

[82] ABC News. Minneapolis man killed in crash on i-94 at highway 280. `http://kstp.com/news/interstate-94-highway-280-fatal-crash/4094871/`, 2016.

[83] D.A. Matthews, S.W. Effler, C.T. Driscoll, S.M. O'Donnell, and C.M. Matthews. Electron budgets for the hypolimnion of a recovering urban lake, 1989-2004: Response to changes in organic carbon deposition and availability of electron acceptors. *Limnology and Oceanography*, pages 743–759, 2008.

[84] Daniel B Neill and Andrew W Moore. Rapid detection of significant spatial clusters. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 256–265. ACM, 2004.

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

[86] Dev Oliver, Shashi Shekhar, Xun Zhou, Emre Eftelioglu, Michael R. Evans, James M. Kang, Qiaodi Zhuang, Renee Laubscher, and Christopher Farah. Significant Linear Hotspot Discovery: A Summary of Results. In *Geographic Information*

Science - 8th International Conference, Vienna, Austria, September 24-26, 2014., pages 284–300, 2014.

[87] X. Li, J. Han, J.G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. *Advances in Spatial and Temporal Databases*, pages 441–459, 2007.

[88] Chris Brunsdon. Computing with spatial trajectories, edited by yu zhengxiao-fang zhou, berlin, springer, 2011, 1st ed. *International Journal of Geographical Information Science*, 27(1):208–209, 2013.

[89] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[90] James MacQueen et al. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14):281–297, 1967.

[91] Dev Oliver, Abdussalam Bannur, James M. Kang, Shashi Shekhar, and Renee Bousselaire. A K-Main Routes Approach to Spatial Network Activity Summarization: A Summary of Results. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 265–272, 2010.

[92] Kevin Buchin, Sergio Cabello, Joachim Gudmundsson, Maarten Löffler, Jun Luo, Günter Rote, Rodrigo I Silveira, Bettina Speckmann, and Thomas Wolle. Finding the most relevant fragments in networks. *J. Graph Algorithms Appl.*, 14(2):307–336, 2010.

[93] S. Shekhar and D.R. Liu. CCAM: A connectivity-clustered access method for networks and network computations. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):102–119, 1997.

[94] Yu Zheng, Huichun Zhang, and Yong Yu. Detecting Collective Anomalies from Multiple Spatio- Temporal Datasets across Different Domains. In *To appear in the 23rd International Conference on Advances in Geographic Information Systems, Seattle, USA, 2015*, 2015.

[95] Shashi Shekhar and Sanjay Chawla. *Spatial databases: a tour*, volume 2003. prentice hall Upper Saddle River, NJ, 2003.

[96] Quantum gis openlayers plugin. `http://plugins.qgis.org/plugins/openlayers_plugin/`, 2014.

[97] Us census bureau tiger/line shapefiles. `http://www.census.gov/geo/maps-data/data/tiger-line.html`, 2016.

[98] Denver crime open dataset, city and county of denver. `https://www.denvergov.org/opendata/dataset/city-and-county-of-denver-crime`, 2016.

[99] Shashi Shekhar, Andrew Fetterer, and Brajesh Goyal. Materialization trade-offs in hierarchical shortest path algorithms. In *International Symposium on Spatial Databases*, pages 94–111. Springer, 1997.

[100] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery. *Knowledge and Data Engineering, IEEE Transactions on*, 24(11):1977–1992, 2012.

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

[102] Surveillance, epidemiology, and end results program. `https://seer.cancer.gov/`, National Cancer Institute, 2017.

[103] Rob T Guerette. *Analyzing crime displacement and diffusion*. US Department of Justice, Office of Community Oriented Policing Services Washington, DC, 2009.

[104] Merrill M Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[105] Ekkehard Köhler, Rolf H Möhring, and Heiko Schilling. Acceleration of shortest path and constrained shortest path computation. In *International Workshop on Experimental and Efficient Algorithms*, pages 126–138. Springer, 2005.

[106] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[107] US Department of Transportation Federal Highway Administration. Roadway safety information analysis: A manual for local rural road owners. `https://safety.fhwa.dot.gov/local_rural/training/fhwasaxx1210/s3.cfm`, 2017.

[108] City of Minneapolis. Pedestrian crash study 2017. `http://www.minneapolismn.gov/pedestrian/data/WCMSP-206913`, 2017.

[109] Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for minimum k-cut. *Algorithmica*, 27(2):198–207, 2000.

[110] Statista The Statistics Portal. Container shipping - statistics and facts. `https://www.statista.com/topics/1367/container-shipping/`.

[111] Quartz Africa Abdi Latif Dahir. A lack of big data is hampering efforts to curb illegal fishing in africa. `https://qz.com/africa/1182628`, 2018.

[112] Devin D Jessee. Tactical means, strategic ends: al qaeda's use of denial and deception. *Terrorism and political violence*, 18(3):367–388, 2006.

[113] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[114] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 35–42. ACM, 2006.

[115] Patrick Laube. *Analysing point motion: Spatio-temporal data mining of geospatial lifelines*. PhD thesis, Universität Zürich, 2005.

[116] Natalia Andrienko and Gennady Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 42(2):117–138, 2007.

[117] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information visualization*, 7(3-4):240–252, 2008.

[118] Torsten Hägerstrand. Space, time and human conditions. 1975.

[119] Kathleen Hornsby and Max J Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):177–194, 2002.

[120] Harvey J Miller. Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information System*, 5(3):287–301, 1991.

[121] Harvey J Miller. A measurement theory for time geography. *Geographical analysis*, 37(1):17–45, 2005.

[122] Andre Salvaro Furtado, Luis Otavio Campos Alvares, Nikos Pelekis, Yannis Theodoridis, and Vania Bogorny. Unveiling movement uncertainty for robust trajectory similarity analysis. *International Journal of Geographical Information Science*, 32(1):140–168, 2018.

[123] Mei-Po Kwan. Gis methods in time-geographic research: Geocomputation and geovisualization of human activity patterns. *Geografiska Annaler: Series B, Human Geography*, 86(4):267–280, 2004.

[124] Hyun-Mi Kim and Mei-Po Kwan. Space-time accessibility measures: A geocomputational algorithm with a focus on the feasible opportunity set and possible activity duration. *Journal of geographical Systems*, 5(1):71–91, 2003.

[125] Bart Kuijpers, Harvey J Miller, Tijs Neutens, and Walied Othman. Anchor uncertainty and space-time prisms on road networks. *International Journal of Geographical Information Science*, 24(8):1223–1248, 2010.

[126] Tijs Neutens, Tim Schwanen, and Frank Witlox. The prism of everyday life: Towards a new research agenda for time geography. *Transport reviews*, 31(1):25–47, 2011.

[127] Dieter Pfoser and Nectaria Tryfona. Capturing fuzziness and uncertainty of spatiotemporal objects. In *East European Conference on Advances in Databases and Information Systems*, pages 112–126. Springer, 2001.

[128] Ying Song and Harvey J Miller. Simulating visit probability distributions within planar space-time prisms. *International Journal of Geographical Information Science*, 28(1):104–125, 2014.

[129] Reynold Cheng, Jinchuan Chen, Mohamed Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 973–982. IEEE, 2008.

[130] Goce Trajcevski, Alok Choudhary, Ouri Wolfson, Li Ye, and Gang Li. Uncertain range queries for necklaces. In *Mobile Data Management (MDM), 11th International Conference on*, pages 199–208. IEEE, 2010.

[131] Edwin H Jacox and Hanan Samet. Spatial join techniques. *ACM Transactions on Database Systems (TODS)*, 32(1):7, 2007.

[132] Jürg Nievergelt and Franco P. Preparata. Plane-sweep algorithms for intersecting geometric figures. *Communications of the ACM*, 25(10):739–747, 1982.

[133] MarineTraffic. Why cannot i see a vessel on the live map? `https://help.marinetraffic.com/hc/en-us/articles/203990958`.