Scalable Data Analytics Techniques for Summarizing Spatial Network-based Observations

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

Dev Oliver

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

Advisor: Professor Shashi Shekhar

May, 2014

Acknowledgements

I thank my advisor, Professor Shashi Shekhar, for his mentorship, support, and guidance throughout my Ph.D. I am truly grateful for the opportunities he has given me and the lessons he has imparted on me. I also thank all the professors who helped me over the years in classes including those who served on my committee: Professor Jaideep Srivastava, Professor Mohamed Mokbel, Professor Sudipto Banerjee, and Professor Vipin Kumar. Thank you for guiding my thesis and helping to shape my overall research.

I thank my collaborators from the National Geospatial-Intelligence Agency for giving me valuable insights into the societal applications of this research and for taking the time to provide feedback on many parts of this thesis. I will miss our regular meetings where we attempted to understand the world a little better through research.

I extend special thanks to Professor Shekhar's Spatial Computing research group. Their beneficial comments have helped me with different aspects of my research and I really apreciate it. I thank Kim Koffolt for taking the time to provide feedback on my papers. I also thank Georganne Tolaas for her help during different milestones of this graduate program.

Lastly, I express my deepest appreciation to my wife Datra and our parents for their unwavering support during this journey.

Abstract

Summarizing spatial network-based observations (SSNO) entails finding a compact description or representation of large spatial or spatio-temporal datasets. For example, transportation planners and engineers may need to identify road segments that pose risks for pedestrians and require redesign. However, SSNO is computationally challenging for the following reasons: (1) There may be a large number of k-subsets of connected components in the network, (2) Patterns may not obey the monotonicity property, and (3) There may be a large number of candidates.

To address the challenge of *the large number of k-subsets of connected components in the network*, we explored the problem of spatial network activity summarization. In spatial network activity summarization (SNAS), we are given a spatial network and a collection of activities (e.g., pedestrian fatality reports, crime reports) and the goal is to find k shortest paths that summarize the activities. SNAS is important for applications where observations occur along linear paths such as roadways, train tracks, etc. Previous work has focused on either geometry or subgraph-based approaches (e.g., only one path), and cannot summarize activities using multiple paths. This work proposes a K-Main Routes (KMR) approach that discovers k shortest paths to summarize activities. To improve performance, KMR uses network Voronoi, divide and conquer, and pruning strategies. Experimental results on synthetic and real data show that KMR with our performance-tuning decisions yields substantial computational savings without reducing summary path coverage. We also present a case study comparing KMR with K-Means on real data.

To address the challenge of *patterns not obeying the monotonicity property*, we studied the problem of geo-referenced time-series summarization. Given a set of regions with activity counts at each time instant (e.g., a listing of countries with number of mass protests or disease cases over time) and a spatial neighbor relation, geo-referenced time-series summarization (GTS) finds k-full trees that maximize activity coverage. GTS has important potential societal applications such as understanding the spread of political unrest, disease, crimes, etc. Previous approaches for spatio-temporal data mining detect anomalous or unusual areas and do not summarize activities. We propose a k-full tree

(kFT) approach for GTS which features an algorithmic refinement (i.e., voronoi partition assignment) that leads to computational savings without affecting result quality. Analytical and experimental results show that the algorithmic refinement substantially reduces computational cost. We also present a case study that shows the output of our approach on Arab Spring data.

To address the challenge of *a large number of candidates*, we tackled the problem of significant linear hotspot discovery (SLHD). Given a spatial network and a collection of activities (e.g., pedestrian fatality reports, crime reports), SLHD finds shortest paths in the spatial network where the concentration of activities is unusually high (i.e., statistically significant). SLHD is important for societal applications in transportation safety, public safety, or public health such as finding paths with significant concentrations of accidents, crimes, or diseases. SaTScan may miss many significant paths since a large fraction of the area bounded by circles for activities on a path will be empty. Previous network-based approaches only consider a small fraction of the network and only one significant network component (e.g., path). We propose novel algorithms for discovering statistically significant linear hotspots using the ideas of likelihood pruning, Monte Carlo speedup, and dynamic segmentation. We present a case study comparing the proposed approach with existing techniques on real data. Experimental results show that the proposed algorithm, with our algorithmic refinements, yields substantial computational savings without reducing result quality.

The work in this thesis is the first step towards understanding the immense challenges and novel applications of summarizing spatial network-based observations for transportation safety, public safety, disaster response, etc. In this thesis, we have formally modeled spatial and spatio-temporal network datasets and have begun to explore new data analytics techniques (e.g., K-Main Routes) that may have many uses in real-world applications. We conclude this thesis by exploring additional data summarization techniques for spatial networks, their computational challenges, and possible directions for future work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many human activities are centered around spatial infrastructure networks, such as transportation, oil/gas-pipelines, and utilities (e.g., water, electricity, telephone). Thus, activity reports such as pedestrian fatalities or crime reports may often use network based location references, e.g., street addresses such as 200 Blizzard Street, Happyville, DQ 94401. The spatial interaction among activities at nearby locations may also be constrained by network connectivity and network distances (e.g., shortest paths along roads or train networks) rather than geometric distances (e.g. Euclidean or Manhattan distances) used in traditional spatial analysis.

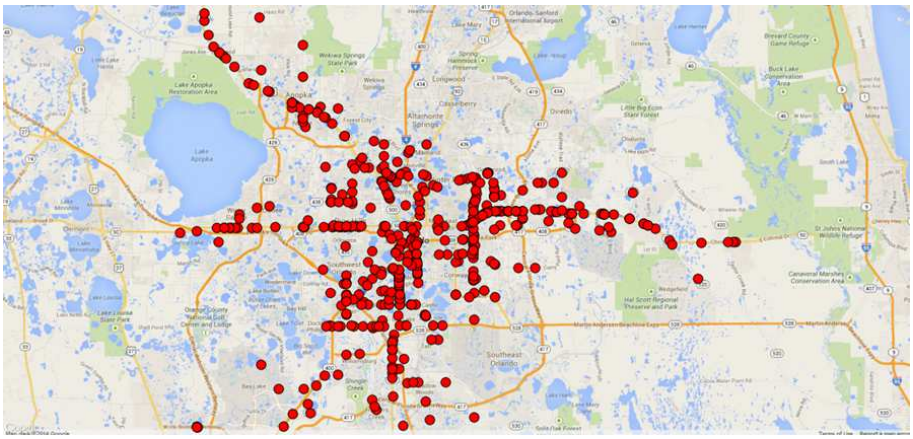


Figure 1.1: Pedestrian fatalities occurring on arterials in Orange County, FL [1]. Activities such as pedestrian fatalities may be constrained by network connectivity and network distances (Best in color)

Figure 1.1 shows an example of 487 pedestrian fatalities occurring on a road network in Orange County, FL from 2001 - 2011 (Orlando is located in Orange County) [1]. As can be seen, the observations are constrained by network connectivity and network distances and are not occurring arbitrarily in space.

1.1 Summarization Framework

Summarizing spatial network-based observations entails finding a compact description or representation of large spatial or spatio-temporal network datasets where the observations (e.g., pedestrian fatality reports, crime reports), are centered around the network. The process typically involves defining a set of groups, finding a representative for each group, and reporting a statistic for each group (e.g., sum, mean, likelihood ratio). These notions differ depending on the genre of the data being summarized.

Table 1.1: Summarization framework for various data genres (Best in Color)

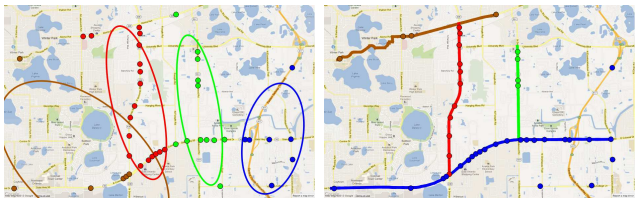
Data Genre (Domain)	Group Definition	Group Representation	Interest Measure
Relational Table (a set of rows)	- A partition of rows - Significant groups of rows	- Attribute values (e.g., age-group)	- aggregate property - max coverage (e.g., count) - significance metric (e.g., p-value, likelihood ratio, confidence interval) - Computational metric - sequential response time
Vector Space	- A partition of vectors - Significant groups of vectors	- A vector, basis function	
Spatial (Euclidean Space)	- A partition of space - Significant groups of sub-space	- points, polygons, ellipses, line-strings	
Spatial Network	- A partition of a network Ch. 2 - Significant groups of sub-networks Ch. 4	- path	
Geo-referenced Time-series (GT)	- A partition of a GT Ch. 3	- ST-full tree	

before thesis thesis

Table 1.1 presents a summarization framework for different genres of data such as relational tables, spatial datasets, spatial networks, and geo-referenced time-series. An example of relational table summarization, is the GROUP BY clause in SQL that is used to group rows having common values to report SQL aggregation functions such as mean and standard deviation [8]. The group definition in this case is a partition of rows and the group representation is distinct values of attributes such as age-groups, citizenship, income-group, etc. Vector space is an alternate view of summarization where the groups may be a partition of vectors or significant groups of vectors, the

group representative may be a vector or a basis function, and the statistic may be an aggregate property (e.g., count, sum, mean) or a significance metric (e.g., p-value, likelihood ratio, confidence interval).

Spatial Euclidean summarization includes heat maps and hotspot analysis. Heat maps provide a graphical representation of data in which individual values contained in a matrix are represented as colors. The group definition for heat maps might include a set of pixels and the group representation may be a subset of these pixels. Hotspots are a special kind of partitioned pattern where objects in hotspot regions have high similarity in comparison to one another and are dissimilar to all the objects outside the hotspot [9]. These spatial summarizations are based on spatial point locations where the group definition is a partition of space and the groups could be represented by points, polygons, ellipses, or line-strings. An example of a spatial summarization is given in Figure 3.1(a) where incidents of crime in a major US city are shown as dots and the spatial (Euclidean) summarization is represented using ten ellipses. The spatial summarization technique used in this case was K-Means [10].



(a) (b)

Figure 1.2: An example of (a) spatial summarization (ellipses) and (b) spatial network summarization (paths).

Spatial network summarization defines groups based on a partition of a graph and represents groups using nodes, paths, trees, etc. For example, the thicker lines in Figure 3.1(b) illustrate a summarization technique that uses linear representatives or paths to represent each partition [11]. Here four paths are used to summarize the incidents of crime (dots) that are on the spatial network (i.e., the road network).

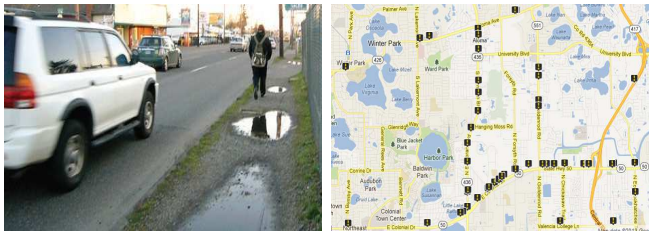
Summarization of the last data genre listed in the table, geo-referenced time-series summarization, defines groups based on a partition of a geo-referenced time series and may represent groups using spatio-temporal (ST)-nodes, ST-paths, or ST-heaviest full

trees.

1.2 Illustrative Application Domains

Two representative application domains are transportation safety and environmental criminology, notably, the areas of preventing pedestrian fatalities and crime analysis.

Preventing Pedestrian Fatalities: According to a recent policy report, more than 47,700 pedestrians were killed in the United States from 2000 and 2009 [2]. 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 [2].



(a) (b)

Figure 1.3: (a) Pedestrian at risk on a road without proper sidewalks [2] (b) Pedestrian fatalities occurring on arterials in Orlando, FL [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 [2]. Daily activities have shifted away from main streets towards higher speed arterials based on the emphasis on traffic movement. 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 4.2(a)). They lack sidewalks, crosswalks (or have crosswalks spaced too far apart), pedestrian refuges, street lighting, and school and public bus shelters [2].

The consequences of this lack of basic infrastructure are fatal. For example, forty percent of fatalities occurred where no crosswalk was available [2]. Figure 4.2(b) shows

a map of pedestrian fatalities that occurred on Orlando roads from 2000 - 2009. Transportation planners and engineers need tools to assist them in identifying which frequently used road segments/stretches pose risks for pedestrians and consequently should be redesigned. Road segments/stretches that pose 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.

Crime Analysis: Environmental criminology is the study of crime, criminality, and victimization as they relate to particular places and to the way that individuals and organizations shape their activities spatially [12]. Environmental criminology is used by law enforcement to develop an understanding of the spatial distributions (e.g., high activity places or hot-spots) of crime activities as well as location-based factors affecting activities using analytical techniques, e.g., crime mapping, and analytical tools such as CrimeStat [13]. Environmental criminology is also used to assist police departments by enabling the design of patrol routes and providing street-based descriptions of crime attractors and generators (e.g., a bar after closing time) [14]. Table 1.2 illustrates real crime report datasets collected by police departments in formats advocated by the US Department of Justice. Crime reports provide locations using symbolic systems (e.g., street addresses, highway-mile markers) as well as numerical systems (e.g., latitude/longitude), which refer to a point.

Table 1.2: Sample crime report data in formats advocated by the US Department of Justice

ID	OFFENCE TYPE	DATE	TIME	ADDRESS
96	Burglary	1/14/2006	1530	16950 GRAND AVE
477	Auto Theft	8/2/2006	2042	7950 W FAIRVIEW 1960
633	Narcotic Drug Laws	11/2/2006	1200	12950 CLEVELAND DR

Environmental criminology has several spatial theories such as Routine Activity Theory (RAT) [15] and Crime Pattern Theory (CPT) [16]. RAT suggests that the location of a crime is related to the criminal's frequently visited areas and CPT extends this theory on a spatial model. Crime is not spread evenly across maps but instead is concentrated in some areas and absent in others [17]. This knowledge is used everyday by people and is seen in the way they avoid some places and seek out others. Police also use this understanding to make decisions about how to allocate scarce resources based

in part on where police demand is highest. Officers are told to be attentive in certain areas but are not given guidance in other areas where crime is scarce [17].

The various areas of concentrated crime (i.e., hotspots) are typically studied in terms of places, neighborhoods, and streets [17]. For places, an explanation as to why crime events occur at specific locations is sought. For neighborhoods, analysts look at large areas and ask questions such as "Which areas are claimed by gangs and which areas are not?". Street-based analysis deals with crimes that occur over small stretched areas such as streets or blocks and examples include street drug dealing, prostitution, and robberies of pedestrians [17]. Figure 1.4 illustrates these various types of hotspots.

Figure 1.4: (a) Types of crime hot spots (b) An example of a street-based (linear) hotspot of crime in a major US city

The National Institute of Justice, which is the research, development and evaluation agency of the U.S. Department of Justice, has pointed out that *"commonly available mapping programs make it easy to identify hot spot places or hot spot areas, but do not make linear hotspots easy to identify. Most clustering algorithms, unfortunately, will show areas of concentration even when a line is the most appropriate dimension"* [17].

1.3 Computational Challenges

Summarizing spatial network-based observations is computationally challenging for the following reasons: (1) There may be a large number of k-subsets of connected components in the network, (2) Patterns may not obey the monotonicity property, and (3) There may be a large number of candidates.

Large number of k-subsets of connected components in the network: Finding

a set of k connected components such as shortest paths is computationally challenging. This is due to the fact that if k connected components are selected from all connected components in a spatial network, there are a large number of possibilities for large k, i.e., $\binom{n}{k}$, where n is the number of connected components. This is because different subsets of k connected components could be overlapping or have the same connected components. For disjoint components, the problem would be relatively less computationally challenging. However, due to overlapping connected components, the general problem remains extremely challenging.

Patterns may not obey the monotonicity property: Given an interest measure to model domain semantics, patterns in the spatial network may not obey the monotonicity property. For example, if the interest measure is to maximize activity coverage, a connected component such as a path or tree with no activity may belong to a larger region with many activities. This makes it more challenging to prune subsets as all connected components may have to be considered. Determining appropriate interest measures for various domains is important in capturing domain semantics. Examples of possible interest measures include maximum activity coverage, minimum average distance, likelihood ratio, etc. Summarization based on maximizing activity coverage works well in domains such as transportation safey because it captures areas with the most activity (e.g., pedestrian fatalities) whereas summarization based on likelihood ratio works well in domains such as epidemiology because it captures regions where the concentration of activities is unusually high (i.e., statistically significant).

Large number of candidates: The search space itself may have potentially large number of candidates. For example, in a transportation planning scenario, there may be as many as 10^{16} shortest paths in a given dataset with hundreds of millions of activities or road network nodes. For large roadmaps such as the 100 million road-segments in the US, this results in prohibitive shortest path computation times.

1.4 Thesis Contributions

The main contributions of this thesis address the three challenges of summarizing spatial network-based observations outlined in the previous section.

First, the challenge of there being a *large number of k-subsets of connected components in the network* was addressed by studying the problem of spatial network activity summarization [18, 11]. We proposed a K-Main Routes (KMR) approach that discovers k shortest paths to summarize activities. KMR generalizes K-means for network space but uses shortest paths instead of ellipses to summarize activities. To improve performance, KMR used network Voronoi, divide and conquer, and pruning strategies. We presented a case study comparing KMR's network-based output (i.e., shortest paths) to geometry-based outputs (e.g., ellipses) on pedestrian fatality data. Experimental results on synthetic and real data showed that KMR with our performance-tuning decisions yielded substantial computational savings without reducing summary path coverage.

Second, we examined the problem of geo-referenced time-series summarization (GTS) to address the challenge where *patterns may not obey the monotonicity property* [19]. We proposed a k-full tree (kFT) approach for GTS which featured an algorithmic refinement that led to computational savings without affecting result quality. The algorithmic refinement is Voronoi partition assignment for partitioning regions. Analytical and experimental results showed that the algorithmic refinement substantially reduced computational cost. We also presented a case study that showed the output of our approach on Arab Spring data.

Finally, addressing the *large number of candidates* challenge was done by exploring the problem of significant linear hotspot discovery [20]. We proposed novel algorithms for discovering statistically significant linear hotspots using the ideas of likelihood pruning, Monte Carlo speedup, and dynamic segmentation. We presented a case study comparing the proposed approach with existing techniques on real data. Experimental results showed that the proposed algorithm, with our algorithmic refinements, yielded substantial computational savings without reducing result quality.

1.5 Thesis Organization

This thesis is organized as follows: In Chapter 2, we tackle the challenge of there being a large number of k-subsets of connected components in the network through the problem of spatial network activity summarization and our proposed K-Main Routes (KMR) algorithm. We examine the challenge where subsets of patterns may not obey

the monotonicity property by studying the problem of geo-referenced time-series summarization and our proposed K-Full Tree (KFT) approach in Chapter 3. Chapter 4 discusses the challenge of a large search space due to the large number of candidates by exploring the significant linear hotspot discovery problem and our proposed approaches. Finally, in Chapter 5, we summarize our findings and identify related areas that remain open for future research.

Chapter 2

A K-Main Routes Approach to Spatial Network Activity Summarization

2.1 Introduction

Spatial network activity summarization (SNAS) has important applications in domains where observations occur along linear paths in the network. Table 2.1 provides examples of such applications where the generative model of observations is inherently linear. For example, transportation planners and engineers may need to identify road segments/stretches that pose risks for pedestrians and require redesign [2]; crime analysts may look for concentrations of crimes along certain streets to guide law enforcement [17]; and hydrologists may try to summarize environmental change on water resources to understand the behavior of river networks and lakes [21].

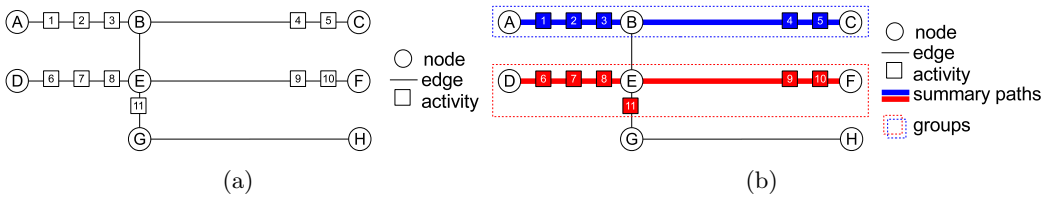
Informally, the SNAS problem can be defined as follows: Given a spatial network, a collection of activities and their locations (e.g., placed on a node or an edge), and a desired number of paths k, find a set of k shortest paths that maximizes the sum of activities on the paths (counting activities that are on overlapping paths only once) and a partitioning of activities across the paths. Depending on the domain, an activity may be the location of a pedestrian fatality, a carjacking, a train accident, etc. SNAS

Table 2.1: Examples of linear generators where observations occur along paths in the spatial network

Linear Generator	Example
Arterial roads	Pedestrian fatalities have largely *"occurred along arterial roadways (i.e., wide, high capacity and high-speed roads) that were dangerous by design, streets engineered for speeding traffic with little or no provision for people on foot, in wheelchairs, or on bicycles"* [2]. Transportation planners and engineers may need to identify frequently used road segments/stretches that pose risks for pedestrians and require redesign.
Crime-prone streets	Crime analysts look for linear concentrations of crime (linear generators and attractors) such as speeding, street drug dealing, drunk driving, etc. to guide law enforcement [17].
Water resource changes in rivers	Environmental engineers may try to summarize environmental change on water resources to understand the behavior of river networks and lakes [21]
Transportation route disaster	*"A snowstorm in Hungary brought drifts 10 feet (3 meters) high and violent gusts of wind, forcing thousands of people to spend the night in their cars or in emergency shelters after being stranded on a major highway"* [22]. Emergency managers may find it useful to summarize the locations of stranded cars on highways (e.g., the M1 highway in Hungary) to better understand how to allocate resources.
Railroads	Transportation officials are interested in understanding railroad accidents (e.g., derailing) to improve safety and reduce cost [23].

assumes that every path is a shortest path because in applications such as transportation planning, the aim is usually to help people to arrive at their destination as fast as possible. Figures 4.1(a) and 4.1(b) illustrate an input and output example of SNAS, respectively. The input consists of eight nodes, seven edges (with edge weights of 1 for simplicity), eleven activities, and $k = 2$, indicating that two routes and groups are desired. Table 2.2 shows the shortest paths for the spatial network in Figure 2.1 and their respective activity coverages (i.e., the sum of activities on each shortest path). The output contains two shortest paths and two groups of activities. The shortest paths are representatives for each group and each shortest path maximizes the activity coverage for the group it represents. For example, route $\langle A, B, C\rangle$ is the representative for the group comprised of activities $1, 2, 3, 4$, and 5, and route $\langle D, E, F\rangle$ is the representative for the group comprised of activities $6, 7, 8, 9, 10$, and 11.

Finding a set of k shortest paths that maximizes the number of activities on selected paths is computationally challenging. This is due to the fact that if k shortest paths are selected from all shortest paths in a spatial network, there are a large number of possibilities for large k, i.e., $\binom{n}{k}$, where n is the number of shortest paths. This is



(a) (b)

Figure 2.1: Example (a) Input and (b) Output of Spatial Network Activity Summarization (Best in color).

because different subsets of k shortest paths could be overlapping or have the same shortest paths. For disjoint paths, the problem would be relatively less computationally challenging. However, due to overlapping paths, the general problem of SNAS is NP-complete and its proof is provided in this chapter (Section 2.4).

Table 2.2: Shortest paths from Figure 2.1 *(Activity Coverage refers to the number of activities on a path)*

SOURCE	SINK	SHORTEST PATH	ACTIVITY COVERAGE	SOURCE	SINK	SHORTEST PATH	ACTIVITY COVERAGE
A	B	$\langle A, B\rangle$	3	C	E	$\langle C, B, E\rangle$	2
A	C	$\langle A, B, C\rangle$	5	C	F	$\langle C, B, E, F\rangle$	4
A	D	$\langle A, B, E, D\rangle$	6	C	G	$\langle C, B, E, G\rangle$	3
A	E	$\langle A, B, E\rangle$	3	C	H	$\langle C, B, E, G, H\rangle$	3
A	F	$\langle A, B, E, F\rangle$	5	D	E	$\langle D, E\rangle$	3
A	G	$\langle A, B, E, G\rangle$	4	D	F	$\langle D, E, F\rangle$	5
A	H	$\langle A, B, E, G, H\rangle$	4	D	G	$\langle D, E, G\rangle$	4
B	C	$\langle B, C\rangle$	2	D	H	$\langle D, E, G, H\rangle$	4
B	D	$\langle B, E, D\rangle$	3	E	F	$\langle E, F\rangle$	2
B	E	$\langle B, E\rangle$	0	E	G	$\langle E, G\rangle$	1
B	F	$\langle B, E, F\rangle$	2	E	H	$\langle E, G, H\rangle$	1
B	G	$\langle B, E, G\rangle$	1	F	G	$\langle F, E, G\rangle$	3
B	H	$\langle B, E, G, H\rangle$	1	F	H	$\langle F, E, G, H\rangle$	3
C	D	$\langle C, B, E, D\rangle$	5	G	H	$\langle G, H\rangle$	0

2.1.1 A Framework for Data Summarization

Data summarization is an important concept in data mining that entails techniques for finding a compact description or representation of a dataset. The process typically involves defining a set of groups, finding a representative for each group, and reporting a statistic for each group (e.g., sum, mean, standard deviation). These notions differ

depending on the genre of the data being summarized. Table 3.1 presents a summarization framework for three genres of data. An example of the first, relational table summarization, is the GROUP BY clause in SQL that is used to group rows having common values to report SQL aggregation functions such as mean and standard deviation. The group definition in this case is a partition of rows and the group representation is distinct values of attributes such as age-group, income-group, etc.

Table 2.3: Summarization framework for various data genres

Data Genre	Group Definition (Partitioning Criteria)	Group Representation Choices	Statistic
Relational Table (a set of rows)	a partition of rows	Distinct values of attributes (e.g., age-group)	sum, count, mean, etc.
Spatial (Euclidean Space)	a partition of space	points, polygons, ellipses, line-strings	sum, count, mean, etc.
Spatial Network (Neighbor Relationship)	a partition of a graph	node, path, tree, subgraph	sum, count, mean, etc.

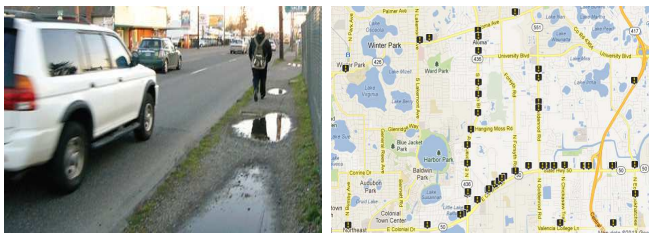
The second genre is spatial Euclidean summarization, which includes heat maps and hotspot analysis. Heat maps provide a graphical representation of data in which individual values contained in a matrix are represented as colors. The group definition for heat maps might include a set of pixels, and the group representation may be a subset of these pixels. Hotspots are a special kind of partitioned pattern where objects in hotspot regions have high similarity in comparison to one another and are dissimilar to all the objects outside the hotspot [9]. These spatial summaries are based on spatial point locations where the group definition is a partition of space and the groups could be represented by points, polygons, ellipses, or line-strings.

In this work we explore the third genre, spatial network summarization, which defines groups based on partitioning a network and may represent groups using nodes, paths, trees, etc.

2.1.2 An Illustrative Application Domain: Preventing Pedestrian Fatalities

To illustrate the applicability of SNAS, we focus on the problem of pedestrian fatalities. According to a recent policy report, more than 47,700 pedestrians were killed in the United States from 2000 and 2009 [2]. 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 [2].



(a) (b)

Figure 2.2: (a) Pedestrian at risk on a road without proper sidewalks [2] (b) Pedestrian fatalities occurring on arterials in Orlando, FL [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 [2]. Daily activities have shifted away from main streets towards higher speed arterials based on the emphasis on traffic movement. 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 4.2(a)). They lack sidewalks, crosswalks (or have crosswalks spaced too far apart), pedestrian refuges, street lighting, and school and public bus shelters [2].

The consequences of this lack of basic infrastructure are fatal. For example, forty percent of fatalities occurred where no crosswalk was available [2]. Figure 4.2(b) shows a map of pedestrian fatalities that occurred on Orlando roads from 2000 - 2009. Transportation planners and engineers need tools to assist them in identifying which frequently used road segments/stretches pose risks for pedestrians and consequently should be redesigned. Road segments/stretches that pose 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 linear concentrations of activities such as pedestrian fatalities in a spatial network.

2.1.3 Related Work

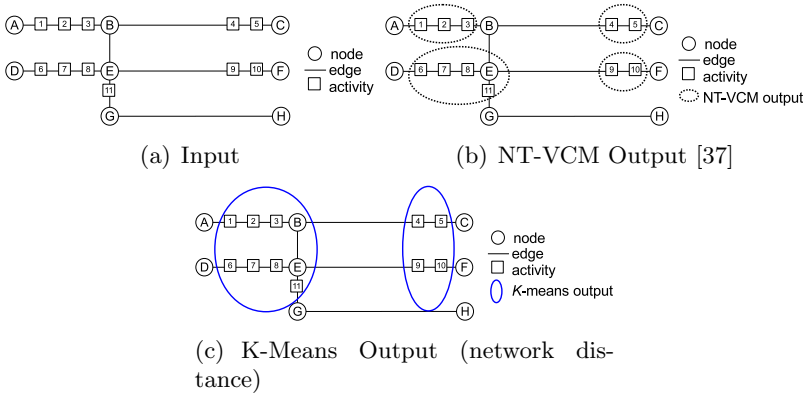
Summarizing activities by grouping is a significant research area in data mining. Previous techniques have generally been geometry-based [10, 24, 25, 26, 27] or network-based [28, 29, 30, 31, 32, 33, 34, 35, 36, 37].

In geometry-based summarization, partitioning of spatial data is based on grouping similar points distributed in planar space where distance is calculated using Euclidean distance, not network distance. Such techniques focus on the discovery of the geometry (e.g., circle, ellipse) of high density regions [17] and include K-Means [10], K-medoid [24, 25], P-median [26] and Nearest Neighbor Hierarchical Clustering [27]. These methods do not consider the underlying spatial network; they group spatial objects that are close in terms of Euclidean distance but not close in terms of network distance. Thus, they may fail to group activities that occur on the same street.

In network-based summarization, spatial objects are grouped using network (e.g., road) distance. Existing methods of network-based summarization such as Mean Streets [28], Maximal Subgraph Finding (MSGF) [29], and Clumping [30, 31, 32, 33, 34, 35, 36, 37] group activities over multiple paths, a single path/subgraph, or no paths at all. Mean Streets [28] finds anomalous streets or routes with unusually high activity levels. It is not designed to summarize activities over k paths because the number of high crime streets returned is always relatively small. MSGF [29] identifies the maximal subgraph (e.g., a single path, $k = 1$) under the constraint of a user specified length and cannot summarize activities when $k > 1$. The Network-Based Variable-Distance Clumping Method (NT-VCM) [37] is an example of the clumping technique [30, 31, 32, 33, 34, 35, 36, 37]. NT-VCM groups activities that are within a certain shortest path distance of each other on the network; in order to run NT-VCM, a distance threshold is needed.

An example output of NT-VCM is shown in Figure 2.3(b). The threshold distance for NT-VCM is the unit distance between activities 8 and 11. However, NT-VCM is not designed to summarize activities using k paths because the grouping of activities is based on the given distance threshold. As such, activities 1, 2, 3, 4, and 5, which occur on the same street would not belong to the same group, given this threshold distance.

Network distance may also be applied to certain geometry-based techniques such as K-Means [10]. Figure 2.3(c) shows an example of the ellipses output by K-Means



(a) Input

(b) NT-VCM Output [37]

(c) K-Means Output (network distance)

Figure 2.3: Two Methods of Summarizing Activities on a Network.

using network distance. The left ellipse groups activities 1, 2, 3, 6, 7, 8, and 11 whereas the right ellipse groups activities 4, 5, 9, and 10. Even when generalized with network distances, these methods output point-based or ellipsoid-based groups, not paths. The objective of network-based K-Means (e.g., minimize the within-cluster sum of squares) is different from that of SNAS (e.g., maximize activity coverage), which explains their difference in output.

Previously, we proposed K-Main Routes (KMR) using inactive node pruning as a heuristic for summarizing activities in a spatial network [11]. We demonstrated the correctness of inactive node pruning, experimentally evaluated KMR, and presented a case study comparing the output of network based with geometry based summarization. However, our previous approach was limited in terms of performance and use of heuristics without proof of NP-completeness.

2.1.4 Contributions

In this chapter, we show that SNAS is NP-complete and propose two additional techniques for improving the performance of KMR: 1) Network Voronoi activity Assignment, which allocates activities to the nearest summary path, and 2) Divide and conquer Summary PAth REcomputation, which calculates the new summary path of each group using only group nodes. Applying these two performance-tuning techniques results in significant computational savings. Specifically, our research contributions are as follows:

- We introduce a summarization framework demonstrating how different genres of data may be summarized.
- We show that SNAS is NP-complete.
- We propose two new techniques for improving the performance of K-Main Routes (KMR): Network Voronoi activity Assignment (NOVA_TKDE) and Divide and conquer Summary PAth REcomputation (D-SPARE_TKDE).
- We analytically demonstrate the correctness of NOVA_TKDE and D-SPARE_TKDE.
- We analyze the computation costs of KMR.
- We present a case study comparing KMR with geometry-based summarization techniques on pedestrian fatality data.
- We test the performance and scalability of KMR using both synthetic and real-world data sets and demonstrate the computational efficiency of our performance-tuning strategies.

2.1.5 Scope and Outline of the chapter

This work focuses on summarizing discrete activity events (e.g., pedestrian fatalities, crime reports) associated with a point on a network. This does not imply that all activities must necessarily be associated with a point in a street. Furthermore, other network properties such as GPS trajectories and traffic densities of road networks [38] are not considered. The objective function used in SNAS is based on maximizing the activity coverage of summary paths, not on minimizing the distance of activities to summary paths. The summary paths are shortest paths but other spatial constraints are not considered (e.g., nearest neighbors) [39]. Additionally, it is assumed that the number of activities on the road network is fixed and does not change over time. A dynamically changing number of activities is presently beyond the scope of this research.

The chapter is organized as follows: Section 4.2 presents the basic concepts and problem statement of SNAS. Section 2.3 explains KMR, NOVA_TKDE, and D-SPARE_TKDE. Section 2.4 details the analytical evaluation. Section 4.5 presents a

case study comparing KMR with other summarization techniques. The experimental evaluation is covered in Section 4.6. Section 4.7 presents a discussion and Section 4.8 concludes the chapter and previews future work.

2.2 Basic Concepts and Problem Statement

This section introduces several key concepts in SNAS and presents a formal problem statement.

2.2.1 Basic Concepts

We define our basic concepts as follows:

Definition 1. *A **spatial network** $G = (N, E)$ consists of a node set N and an edge set E, where each element u in N is associated with a pair of real numbers (x, y) representing the spatial location of the node in a Euclidean plane [40]. Edge set E is a subset of the cross product $N \times N$. Each element $e = (u, v)$ in E is an edge that joins node u to node v.*

An example of a spatial network is shown in Figure 2.1(a). In the figure, 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 $e \in E$ or one node $n \in N$.*

In Figure 2.1(a), activities are represented as squares. 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; and in disaster response an activity may be the location of a request for relief supplies.

Definition 3. *A **summary path set** $\hat{P}$ is a collection of summary paths where each path $p_i \in \hat{P}$ is a shortest path. A **summary path** imposes a partitioning on an activity set A such that network distance$(a, p_i) \leq$ network distance(a, p_j) $\forall p_j \in \hat{P},\ \forall a \in A$.*

Figure 2.1(b) shows two summary paths $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$. Activities $1, 2, 3, 4$, and 5 form a partition around $\langle A, B, C\rangle$ because they are closer to $\langle A, B, C\rangle$ whereas

activities $6, 7, 8, 9, 10$ and 11 form a partition around $\langle D, E, F\rangle$ because they are closer to $\langle D, E, F\rangle$. Here the network distance between an activity and a path, i.e., *network distance*(a, p_i), is the network distance between a and the closest node in p_i.

Definition 4. *The* ***activity coverage*** $AC(p)$ ***of a path*** p *is the sum of activities having network distance = 0 from an edge* $e \in p$*. The* ***activity coverage*** $AC(P)$ ***of a set of paths*** P *is the sum of activities across individual paths* p_i *in set* P*, having network distance = 0 from each edge* $e \in P$*, counting activities that are covered several times only once. If two paths share an edge, the activities on that edge are only counted once.*

For example, in Figure 2.1(a) the activity coverage of the shortest path from node A to node B is 3 because there are 3 activities occurring on that path. Likewise, the activity coverage of the shortest path from node D to node F is 5 because there are 5 activities occurring on $\langle D, E, F\rangle$. If the set of paths are $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$, then the activity coverage is 10 because there are 10 activities on all the edges of the paths in P. If the set of paths are $\langle A, B, C\rangle$ and $\langle A, B\rangle$, then the activity coverage is 5. Note that the activities on edge AB are only counted once even though AB is an edge in both paths.

2.2.2 Problem Statement

The problem of spatial network activity summarization (SNAS) can be expressed as follows:

Given:

1. A spatial network $G = (N, E)$ with weight function $w(u, v) \geq 0$ for each edge $e = (u, v) \in E$ (e.g., network distance),
2. A set of activities A and their locations (e.g., a node or an edge),
3. A desired number of summary paths, k, where $k \geq 1$.

Find:

1. A summary path set of size k,

2. A partitioning of activities across these summary paths.

Objective: Maximize the activity coverage of each summary path for the group it represents.

Constraints:

1. Each summary path is a shortest path between its end-nodes,

2. Each activity $a \in A$ is associated with only one edge $e \in E$.

The spatial network input for SNAS is defined in Definition 8. The activities input are objects of interest in the spatial network such as the locations of pedestrian fatalities. The k input represents the desired number of summary paths. The output for SNAS is a summary path set of size k and a partitioning of activities across the paths. The summary paths are representatives for each group and each summary path maximizes the activity coverage for the group it represents. The optimal solution for SNAS may not be unique and this is shown in lemma 1.

Example. The network in Figure 4.1(a) can be viewed as a road network, composed of streets (edges) and intersections (nodes) with eleven activities (squares). The aim is to find two routes and two groups of activities; the routes are representatives for each of the groups. In a transportation planning scenario, identifying such routes would guide street redesign efforts to reduce the risk of pedestrian fatalities (e.g., adding sidewalks, crosswalks, pedestrian refuges, street lighting, etc.). In Figure 4.1(b), route $\langle A, B, C\rangle$ is the representative for the group comprised of activities $1, 2, 3, 4$, and 5; and route $\langle D, E, F\rangle$ is the representative for the group comprised of activities $6, 7, 8, 9, 10$ and 11.

2.3 Spatial Network Activity Summarization

This section describes the computational structure of SNAS. It also describes the K-Main Routes (KMR) algorithm and its performance-tuning decisions Network Voronoi activity Assignment, Divide and conquer Summary PAth REcomputation, and Inactive Node Pruning.

2.3.1 Computational Structure of SNAS

In SNAS, the optimal solution may not be unique. Additionally, among the optimal solutions there are some where every path starts and ends at active nodes. These properties are formally shown via Lemmas 1 and 2.

Definition 5. *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 or a node that has one or more activities, or both. An **inactive node** is a node that is not joined by any active edges.*

Edges AB and BC in Figure 2.1(a) are active edges because they each have at least one activity and nodes A, B, C, D, E, F, and G are all active nodes because they are all joined by active edges. By contrast, Node H is an inactive node because it is not joined by any active edges.

Lemma 1. *The optimal solution for SNAS may not be unique.*

Proof. There may be multiple solutions for different values of k. For example, given $k = 1$ in Figure 2.1(a) where all eleven activities are members of the same group, the summary path could be $\langle A, B, E, D\rangle$ or $\langle D, E, B, A\rangle$, since both these paths have a maximum activity coverage of 6 based on the one group. Given $k = 2$ and the groups shown in Figure 4.1(b), the summary paths could be $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$ or $\langle C, B, A\rangle$ and $\langle F, E, D\rangle$ as both sets of paths have a maximum activity coverage of 11 based on their respective groups. □

Lemma 2. *Among the optimal solutions for SNAS, there exist optimal solutions where every path starts and ends at active nodes.*

Proof. Let's begin with an arbitrary optimal solution. Let p be a shortest path that starts or ends with inactive nodes. If inactive nodes that start or end p are removed such that p starts and ends with active nodes, the resulting subpath p' is still optimal in terms of activity coverage, because no active edges were removed. p' is also still a shortest path due to the optimal substructure of shortest paths wherein subpaths of shortest paths are shortest paths [41]. In other words, eliminating inactive nodes from the beginning and end of a shortest path does not reduce coverage and does not split the path. KMR takes advantage of this property to achieve computational savings. □

2.3.2 K-Main Routes Algorithm

Algorithm 1 presents the pseudocode for the proposed K-Main Routes (KMR) approach. The basic structure of KMR resembles that of K-Means [10] in terms of selecting initial seeds, forming k groups, and updating the representative of each group until the assignments no longer change. Line 1 of Algorithm 1 selects all shortest paths P that start and end with active nodes (inactive node pruning). Next, k paths from P are selected as initial summary paths, which are the "seeds" for KMR (line 2). The algorithm then proceeds in two main phases. First, it forms k groups by assigning each activity to its closest summary path (line 4). Then, it updates the summary path of each group by calculating the shortest path that maximizes activity coverage (line 5). Assigning and updating repeat until the summary paths no longer change and the final summary paths and groups are returned (line 8).

Algorithm 1 K-Main Routes (KMR) Algorithm

Input:

1) a spatial network $G = (N, E)$,
2) a set of activities A,
3) a number of routes k,
4) mode1 $\in \{naive, NOVA\_TKDE\}$,
5) mode2 $\in \{naive, D\text{-}SPARE\_TKDE\}$

Output:

A summary path set of size k and a partitioning of activities across these summary paths, where the objective is to maximize the activity coverage of each summary path for the group it represents.

Algorithm:

1: $P \leftarrow$ shortest paths between active nodes of G
2: $\hat{P} \leftarrow k$ summary paths $\in P$; $stableGroups \leftarrow false$;
3: **while** not $stableGroups$ **do**
4: **Phase 1**: $currentGroups \leftarrow AssignActivities\text{-}ToSummaryPaths(G, A, k, \hat{P}, mode1)$
5: **Phase 2**: $\hat{P}' \leftarrow RecomputeSummaryPaths(G, A, k, currentGroups, mode2)$
6: **if** $\hat{P} = \hat{P}'$ **then** $stableGroups \leftarrow true$
7: $\hat{P} \leftarrow \hat{P}'$
8: **return** $currentGroups$

KMR Example: Figure 4.5 shows an example execution trace of KMR. The spatial

network shown has eight nodes, seven edges, and eleven activities. For illustration purposes, we choose $\langle A, B\rangle$ and $\langle D, E\rangle$ as initial summary paths, $k = 2$, and set all edge weights to 1.

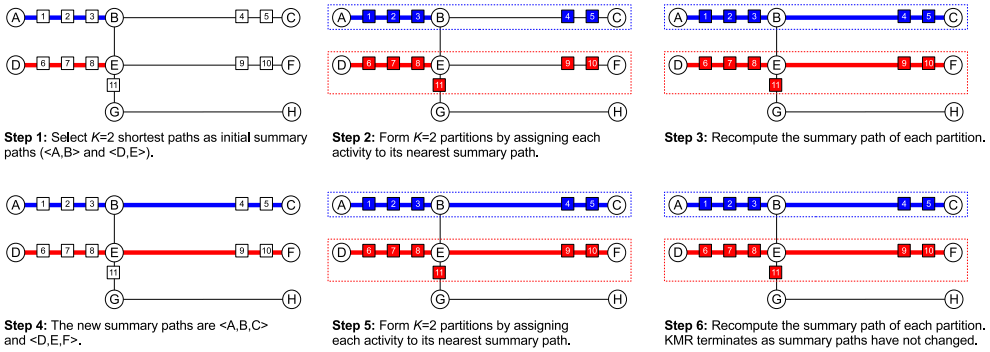


Figure 2.4: Execution trace of K-Main Routes (KMR). Circles represent nodes, lines represent edges, and squares represent activities (Best in color).

In step 2 of Figure 4.5, two groups are formed by assigning each activity to its closest summary path. In this example, activities 1, 2, 3, 4, and 5 are assigned to the summary path $\langle A, B\rangle$, and activities 6, 7, 8, 9, 10 and 11 are assigned to the summary path $\langle D, E\rangle$. The groups are highlighted with dashed lines. In cases where the distance between an activity and several summary paths is equal, the activity is assigned to one of those summary paths at random.

Once groups are formed, the summary paths of each group have to be recalculated, as shown in step 3. In this case, the new summary paths (shown in step 4) are $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$. These summary paths are chosen because they further maximize the activity coverage. In other words, based on the activities of each group, summary paths $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$ have maximum activity coverage.

Step 5 repeats step 2 using the new summary paths $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$. The groups formed in this case are indicated by the dashed lines. Step 6 involves another recalculation of the summary paths to further maximize the activity coverage. The summary paths that are recalculated do not change and as a result, the algorithm terminates. We now discuss each phase of KMR in detail.

Phase 1: Assign activities to nearest summary paths

In this phase, k groups are formed by assigning each activity to its closest summary path. Algorithm 2 presents the pseudocode for the activity assignment algorithm which has two modes: naive and NOVA_TKDE. The naive mode enumerates all the distances between every activity and summary path, and then assigns each activity to its closest summary path. NOVA_TKDE, by contrast, avoids the enumeration that is done in the naive mode while still providing correct results.

Performance-Tuning for Phase 1: The Network Voronoi activity Assignment (NOVA_TKDE) technique is a faster way of assigning activities to the closest summary path. Consider a virtual node, V, that is connected to every node of all summary paths by edges of weight zero. The basic idea is to calculate the distance from V to all active nodes and discover the closest summary path to each activity. The shortest path from V to each activity a will go through a node in the summary path that is closest to a.

NOVA_TKDE starts by initializing all the relevant data structures. Line 5 of Algorithm 2 initializes the virtual node V connected to each node of all summary paths by edges of weight 0. Line 6 initializes the *Open* list and T_{nodes} to V and the *Closed* and $T_{activities}$ to the empty set. NOVA_TKDE then expands every node in the *Open* list based on how close it is to the summary paths; closer nodes get expanded first. Once a node n is expanded, it is moved to the *Closed* list (line 8). Next, each of $n's$ neighbors $x_i \notin Closed$ is examined, and T_{nodes} is updated with x_i's *distance* and *sp* information, where $x_i.distance$ is the network distance of x_i from the nearest summary path, and $x_i.sp$ is x_i's assigned summary path. $x_i.distance$ is calculated by adding $n.distance$ to the distance of edge (n, x_i) (line 9). If x_i is not in *Open*, it is then added to the *Open* list (line 10).

Once NOVA_TKDE finds activities on an edge connecting node n to a summary path, it records the activity distance to that summary path. Every activity a_i that is on edge (n, x_i) is examined, and $T_{activities}$ is updated with a_i's *distance* and *sp* information, where $a_i.distance$ is the network distance of a_i from the nearest summary path (based on n), and $a_i.sp$ is the assigned summary path of a_i. Next, an activity is assigned to a summary path (line 12). If the activity was previously assigned to another summary path, it is removed from that path before being assigned to the new summary path. Once all active nodes have been added to the *Closed* list, or the *Open* list is empty,

NOVA_TKDE's main loop is stopped (line 13). If unassigned activities remain due to no connectivity to summary paths, these activities are randomly assigned to any summary path $\in \hat{P}$ (line 14). NOVA_TKDE then returns the current groups and terminates (line 15).

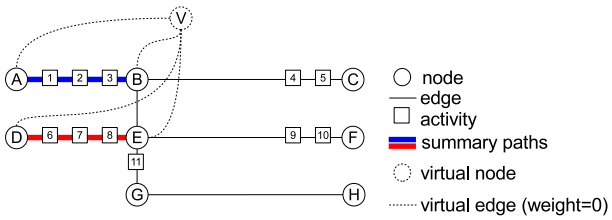


Figure 2.5: An example of NOVA_TKDE. Activity 10 gets assigned to summary path $\langle D, E\rangle$ because the shortest path $\langle V, E, F\rangle$ from V to activity 10 goes through node E of summary path $\langle D, E\rangle$ (Best in color).

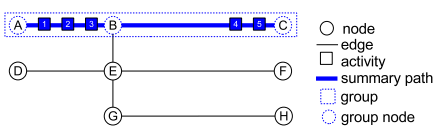
An example of NOVA_TKDE activity assignment is shown in Figure 2.5. Virtual node V is connected by zero weight edges to nodes A and B of summary path $\langle A, B\rangle$ and nodes D and E of summary path $\langle D, E\rangle$ (Algorithm 2, line 5). Activity 10 gets assigned to summary path $\langle D, E\rangle$ because the shortest path $\langle V, E, F\rangle$ from V to activity 10 goes through node E of summary path $\langle D, E\rangle$. Similarly, activity 5 gets assigned to summary path $\langle A, B\rangle$ because the shortest path $\langle V, B, C\rangle$ from V to activity 5 goes through node B of summary path $\langle A, B\rangle$.

Phase 2: Recompute summary paths

In phase 2, the summary path of each group is recomputed so as to further maximize activity coverage (i.e., the number of activities covered by the set of paths). Algorithm 3 presents the pseudocode, which has two modes: naive and D-SPARE_TKDE. The naive mode enumerates the shortest paths between all active nodes in the spatial network while D-SPARE_TKDE considers only the set of shortest paths between the active nodes of a group, which gives the correct results.

Performance-Tuning for Phase 2: The Divide and Conquer Summary PAth REcomputation (D-SPARE_TKDE) technique chooses the summary path of each group with maximum activity coverage but only considers the set of shortest paths between the nodes of a given group (Algorithm 3, lines 8-9). D-SPARE_TKDE assumes rich

connectivity and otherwise may return a summary path going outside a given fragment. If $maxPath$ is null after looking at the shortest paths in a given group, $c_i \in currentGroups$, the shortest path which has the maximum activity coverage based on the activities in c_i is selected as $maxPath$ (lines 10-12). $maxPath$ is then added to $\hat{P}$ as the new summary path for c_i (line 13). Once all groups $c_i \in currentGroups$ have been considered, $\hat{P}$, which contains the summary paths with maximum activity coverage for each group, is returned.



Shortest paths considered:

Source	Sink	Shortest Path	AC
A	B	<A,B>	3
A	C	<A,B,C>	5
B	A	<B,A>	3
B	C	<B,C>	2
C	B	<C,B>	2
C	A	<C,B,A>	5

Figure 2.6: An example of D-SPARE_TKDE. Only shortest paths between nodes in the group are considered when choosing the new summary path which maximizes activity coverage. In this case, summary paths $\langle A, B, C\rangle$ and $\langle C, B, A\rangle$ both maximize activity coverage based on activities $1, 2, 3, 4$, and 5 so D-SPARE_TKDE will choose one of these summary paths as the new representative for this group (Best in color).

Figure 2.6 shows an example of D-SPARE_TKDE where the given group consists of activities $1, 2, 3, 4$, and 5, and summary path $\langle A, B\rangle$. The goal of D-SPARE_TKDE is to choose a new summary path that maximizes activity coverage for every group based on the activities of each group. In this case, summary paths $\langle A, B, C\rangle$ and $\langle C, B, A\rangle$ both maximize activity coverage based on activities $1, 2, 3, 4$, and 5, so D-SPARE_TKDE will choose one of these summary paths as the new representative for this group.

KMR with Inactive Node Pruning Performance-Tuning: We also applied a pruning strategy to K-Main Routes to improve its performance. Rather than checking all shortest paths, inactive node pruning considers only paths between active nodes, thereby reducing the total number of paths considered (Algorithm 1, line 1). For example, Figure 2.1(a) shows a spatial network with eight nodes, seven edges, and eleven activities. The active nodes in this network are A, B, C, D, E, F, and G. Without inactive node pruning, the number of shortest paths considered would be 56 because there are eight nodes, and the shortest path between each node and every other node is considered. With inactive node pruning, the number becomes 42, as only the shortest paths between the seven active nodes are considered.

2.4 Theoretical Analysis

In this section, we present a proof of NP-Completeness for spatial network activity summarization. We also present proofs of correctness for our proposed performance-tuning techniques.

2.4.1 Proof of NP-Completeness

For simplicity, we begin by defining a generalized, decision version of SNAS where the set of paths might be arbitrary and show this problem to be NP-complete. We then present a proof sketch showing that the decision version of SNAS with shortest paths is also NP-complete.

Definition 6. *Decision version of SNAS:*

INSTANCE: *A spatial network $G = (N, E)$ with weight function $w(u, v) \geq 0$ for each edge $e = (u, v) \in E$, a set of activities A and their locations, a set of paths P, a desired number of routes k, and a bound $B \in Z^+$, where Z^+ denotes the set of positive integers.*

QUESTION: *Does P contain a cardinality k subset P' of P, i.e., a subset $P' \subseteq P$ with $|P'| = k$ and $AC(P') \geq B$, where $AC(P')$ denotes the activity coverage of P'?*

Theorem 1. *SNAS is NP-complete.*

Proof. The process of devising an NP-completeness proof for a decision problem Π consists of the following four steps [42]:

1. showing that Π is in NP,
2. selecting a known NP-complete problem Π',
3. constructing a transformation f from Π to Π', and
4. proving that f is a (polynomial) transformation

In step 1, to show that SNAS $\in$ NP, assume that a certificate and a number B are given. The certificate consists of a spatial network $G = (N, E)$ with weight function $w(u, v) \geq 0$ for each edge $e_i = (u, v) \in E$, a set of activities A, a set of paths P, a

desired number of routes k, and a cardinality k subset P' of P, i.e., a subset $P' \subseteq P$ with $|P'| = k$. We can then verify in polynomial time whether $AC(P') \geq B$ because $AC(P')$ involves counting the number of activities in P'.

Step 2 selects the Maximum Coverage problem [43] as a known NP-complete problem Π'. Although the optimization version of Maximum Coverage [43] is known to be NP-Hard, its decision version is NP-Complete. The decision version is specified as follows:

INSTANCE: A number k and a collection of sets $S = S_1, S_2, \ldots, S_m$, where $S_i \subseteq \{l_1, l_2, \ldots, l_n\}$, and a bound $B \in Z^+$, where Z^+ denotes the positive integers.

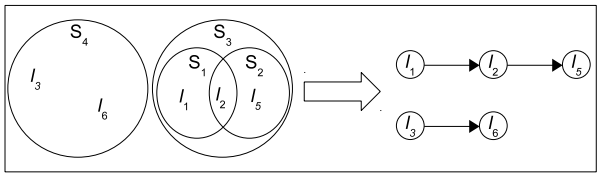
QUESTION: Does S contain a subset $S' \subseteq S$ of sets such that $|S'| \leq k$ and the number of covered elements $\left| \bigcup_{S_i \in S'} S_i \right| \geq B$?

Steps 3 and 4 construct a transformation f from Π to Π' and prove that f is a (polynomial) transformation. The reduction entails a polynomial time transformation of the input of Maximum Coverage to the input of SNAS followed by a polynomial time transformation of the output of SNAS to the output of Maximum Coverage. The input of Maximum Coverage may be transformed to the input of SNAS using the following steps:

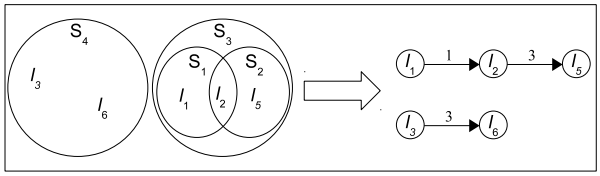
1. Impose a total order TO on n elements in $L = \{l_1, l_2, ..., l_n\}$.
2. Convert each element in L into a node with one activity.
3. Convert each set S_i to a path P_i .
    - Add edge (l_j, l_{j+1}) $\forall$ $j \in 1...|S_i|$.

The transformation computation time is dominated by the polynomial step of sorting the elements in S_i using TO. Thus it is easy to see that the entire transformation is indeed polynomial. Consider an instance of the Maximum Coverage problem as shown in Figure 2.7(a) where $L = \{l_1, l_2, l_3, l_5, l_6\}$, $k = 2$, $S_1 = \{l_1, l_2\}$, $S_2 = \{l_2, l_3\}$, $S_3 = \{l_1, l_2, l_3\}$, and $S_4 = \{l_5, l_6\}$. The resulting instance of SNAS would thus be $P = \{(l_1 \rightarrow l_2), (l_2 \rightarrow l_3), (l_1 \rightarrow l_2 \rightarrow l_3), (l_5 \rightarrow l_6)\}$, $k = 2$, $A = \{a_1, a_2, a_3, a_5, a_6\}$, and $ActivityNode = \{a_1(l_1), a_2(l_2), a_3(l_3), a_5(l_5), a_6(l_6)\}$.

Next, we convert the instance of SNAS output to an instance of Maximum Coverage output. The transformation, which is also polynomial, is as follows:



(a)



(b)

Figure 2.7: SNAS instance resulting from Maximum Coverage instance for (a) arbitrary paths and (b) shortest paths.

- For each k path P_i produced by SNAS, convert the activities on the path into elements and form a set S_i.

The candidate solutions for the instance of SNAS shown in Figure 2.7(a) are $(l_1 \rightarrow l_2 \rightarrow l_3)$ and $(l_5 \rightarrow l_6)$. The resulting instance of the Maximum Coverage output would be $S_1 = \{l_1, l_2, l_3\}$ and $S_2 = \{l_5, l_6\}$.

The reduction of Maximum Coverage to SNAS is a polynomial time reduction since the input of Maximum Coverage can be reduced to SNAS in polynomial time, and the output of SNAS can be reduced to Maximum Coverage in polynomial time.

Since SNAS belongs to the class of NP and a known NP-complete problem is reduced to it, the decision version of SNAS is NP-complete.

We now present a proof sketch that the decision version of SNAS where the set of paths are shortest paths is also NP-complete. We follow the same construction as before but assign a cost for the edge (l_i, l_j) to be $j - i$ (Figure 2.7(b)). This step ensures that all paths generated by construction are shortest paths. □

2.4.2 Correctness of Performance-Tuning Decisions

We prove the correctness of inactive node pruning, NOVA_TKDE, and D-SPARE_TKDE as follows:

Theorem 2. *Inactive node pruning is a correct filtering method.*

Proof. A filtering method is considered correct if it does not exclude all optimal solutions. By Lemma 2, among the optimal solutions for SNAS, there exist optimal solutions where every path starts and ends at active nodes. Since inactive node pruning does not prune out any path that starts and ends at active nodes, no such optimal solutions will be pruned. Thus, inactive node pruning is correct. □

Theorem 3. *Network Voronoi activity Assignment (NOVA_TKDE) is a correct activity assignment method, i.e., it assigns every activity to the nearest summary path.*

Proof. An activity assignment method is considered correct if it assigns every activity to its nearest summary path. Let each activity be located at a particular node $n \in N$ and let a virtual source node V be added to $G = (N, E)$ such that V is connected by an edge of zero weight to each node in the set of summary paths, $\hat{P}$. The correctness of NOVA_TKDE can be argued in a similar fashion to that of Dijkstra's algorithm [41], where each active node containing activity a is connected to V via the nearest node $n \in \hat{P}$. Since a is assigned to the summary path $p \in \hat{P}$ containing node n, it is assigned to the nearest summary path. Thus, NOVA_TKDE is correct. □

Theorem 4. *Divide and Conquer Summary Path Recomputation (D-SPARE_TKDE) is a correct summary path recomputation method, i.e., it returns a shortest path with maximum activity coverage for a given group $c \in C$.*

Proof. A summary path recomputation method is considered correct if it returns a shortest path with maximum activity coverage for a given group $c \in C$. In recomputing a summary path $p \in c$, all shortest paths between nodes in a given group c are considered. Thus, the optimal solution, i.e., the shortest path with maximum activity coverage in c, will not be filtered and will be returned. Thus, D-SPARE_TKDE is correct. □

Lemma 3. *AssignActivitiesToSummaryPaths terminates on all inputs*

Proof. AssignActivitiesToSummaryPaths has two modes: naive and NOVA_TKDE. The naive mode terminates on all inputs because there are a finite number of distances considered in assigning activities to the nearest summary path since there are a finite number of activities and summary paths. If the distances between an activity a and

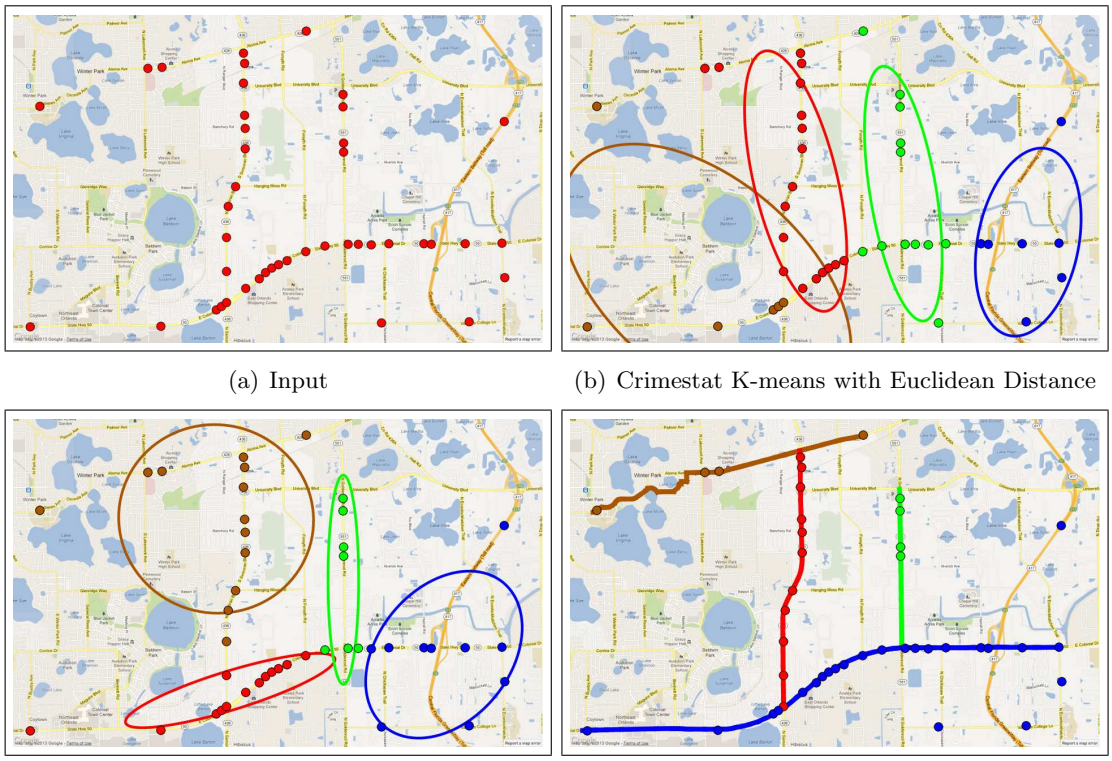
several summary paths are equal, then a gets assigned to one of the summary paths at random. Therefore, even if no path exists between an activity and any summary path, the distances between that activity and all summary paths will all be equal to infinity and the activity will be assigned to one of the summary paths at random. In the NOVA_TKDE mode, each node from the *Open* list that is examined gets moved to the *Closed* list, and the main loop ends when either all active nodes are moved to the *Closed* list or the *Open* list is empty. In the case where no path exists between an activity and any summary path, the activity will be assigned to one of the summary paths at random. Since the number of nodes, summary paths, and activities is finite, NOVA_TKDE will also terminate on all inputs. Since both naive and NOVA_TKDE terminate on all inputs, AssignActivitiesToSummaryPaths also terminates on all inputs. □

Lemma 4. *RecomputeSummaryPaths terminates on all inputs*

Proof. RecomputeSummaryPaths has two modes: naive and D-SPARE_TKDE. The naive mode terminates on all inputs because it considers a finite set of shortest paths between active nodes and returns the path with maximum activity coverage.
D-SPARE_TKDE also terminates on all inputs because it considers a subset of the finite set of shortest paths considered in the naive mode (i.e. shortest paths between the active nodes of a group). Since both naive and D-SPARE_TKDE terminate on all inputs, RecomputeSummaryPaths also terminates on all inputs. □

Lemma 5. *KMR terminates on all inputs.*

Proof. KMR terminates when either global or local maxima are reached. Termination is guaranteed because activity coverage must increase for the next iteration to happen and the activity coverage of the currently selected paths is bounded by the total number of input activities. Each iteration has two phases, each of which is guaranteed to terminate, i.e., AssignActivitiesToSummaryPaths terminates on all inputs (Lemma 3) and RecomputeSummaryPaths terminates on all inputs (Lemma 4). Since each iteration is guaranteed to terminate, the activity coverage must increase for the next iteration to occur, and there are a finite number of activities, KMR terminates on all inputs. □



(a) Input (b) Crimestat K-means with Euclidean Distance

(c) Crimestat K-means with Network Distance (d) KMR

Figure 2.8: Comparing KMR and Crimestat K-means output for $k = 4$ on pedestrian fatality data from Orlando, FL [1] (Best in color).

2.5 Case Study

We conducted a qualitative evaluation of KMR comparing its output with the output of Crimestat K-means [10, 44] on a real pedestrian fatality data set [1], shown in Figure 4.9(a). The input consists of 43 pedestrian fatalities (represented as dots) in Orlando, Florida occurring between 2000 and 2009. As we've explained, KMR uses paths and network distance to group activities on a spatial network. By contrast, in geometry-based summarization, the partitioning of spatial data is based on grouping similar points distributed in planar space where the distance is calculated using Euclidean distance. Such techniques focus on the discovery of the geometry (e.g., circle, ellipse) of high density regions [17] and include K-means [10, 45, 46, 47, 48, 49], K-medoid [24, 25], P-median [26] and Nearest Neighbor Hierarchical Clustering [27] algorithms.

When evaluating the techniques, we consider two outputs: 1) the groups (represented by colors) and 2) the representatives of each group (e.g., paths or ellipses). As noted earlier, pedestrian fatalities usually occur on streets, particularly along arterial roadways [2]. Thus this activity can be said to have a linear generator. However, the results generated by Crimestat do not capture this. From Figure 2.8(b), it is clear that the ellipse-based output is meant for areas, not streets. Results with Crimestat K-Means using network distance are no better. Although the red ellipse in Figure 2.8(c) is aligned with a part of the arterial road, not all the activities on this arterial are captured. Instead, the activities that occur on the road towards the bottom of the figure are split among the red, green, and blue groups. In contrast, the groups of activities in KMR fully capture the activities on the arterial roads (Figure 2.8(d)). For example, the blue group and summary path capture the activities on the arterial road that were split across three groups in network-based K-Means. Furthermore, the path-based representatives in the figure make sense in this context due to the inherently linear nature of the activities.

2.6 Experimental Evaluation

The goal of our experiments was to evaluate KMR with and without performance-tuning. Scalability was evaluated by varying and observing the effect of four workload parameters: nodes, activities, routes, and active node ratio.

Definition 7. *The **active node ratio**, ANR, is the ratio of active nodes to all nodes.*

The active node ratio in Figure 2.1(a) is 7/8, i.e., the total number of active nodes, 7, divided by the total number of nodes, 8.

For each workload experiment, we ran seven versions of KMR:

- KMR with NOVA_TKDE only (KMR_V)
- KMR with D-SPARE_TKDE only (KMR_D)
- KMR with both NOVA_TKDE and D-SPARE_TKDE (KMR_{VD})
- KMR with inactive node pruning only (KMR_I) [11]

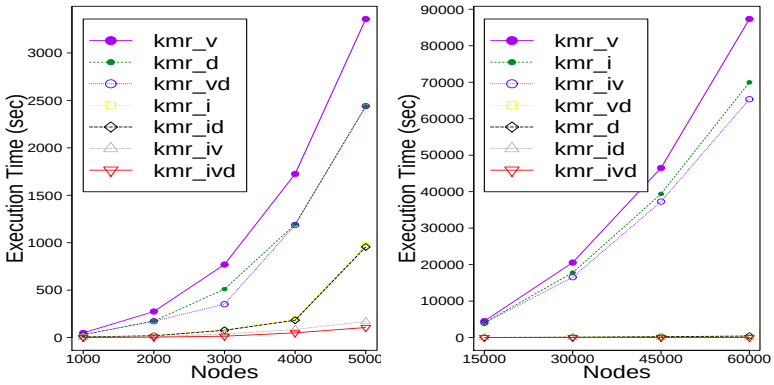
- KMR with both inactive node pruning and NOVA_TKDE (KMR_{IV})
- KMR with both inactive node pruning and D-SPARE_TKDE (KMR_{ID})
- KMR with all three performance-tuning decisions (KMR_{IVD})

All experiments were performed on a Mac Pro with a 2 x Xeon Quad Core 2.26 GHz processor and 16 GB RAM. In all experiments, the summary paths were initialized to the top-k edge-disjoint shortest paths.

2.6.1 Experiment Data Sets

Our experiments were performed on both synthetic and real-world data. Synthetic data were used to evaluate the scalability of KMR and to vary key parameters such as the active node ratio, that could not be varied in the real-world data set. The synthetic data set was a road network generated using the US Census Bureau's 2010 TIGER/Line Shapefiles for Hennepin County, Minnesota [50]; a number between 0 and 100, representing activities, was associated with each street.

The real-world data set, obtained from the Fatality Analysis Reporting System (FARS) encyclopedia [1], is a geospatial and temporal data set describing pedestrian fatalities in Orange County, FL (which includes Orlando), from 2001 to 2011. Every activity was associated with its closest street (Euclidean distance) in the road network, obtained from the US Census Bureau's TIGER/Line Shapefiles for Orange County [50].



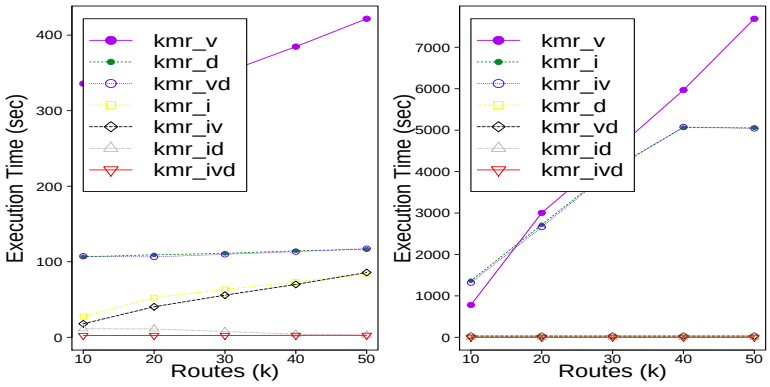
(a) Synthetic data (b) Real data

Figure 2.9: Scalability of KMR with increasing number of nodes

2.6.2 Experimental Results

Effect of the Number of Nodes

Both synthetic and real-world data sets were used to observe the effect of increasing the number of nodes on execution time. For the synthetic data, the number of routes k was set to 2, the number of activities was set to $1,200$, and the active node ratio was set to 0.2. For the real-world data, the number of routes k was set to 30, the number of activities was 602 for $60,000$ nodes, and the active node ratio for the $60,000$ nodes was 0.007. Figure 2.9 gives the execution times for both data sets. As can be seen, KMR_{IVD} is the fastest. Computational savings increases as the number of nodes increases due to NOVA_TKDE, D-SPARE_TKDE, and inactive node pruning.



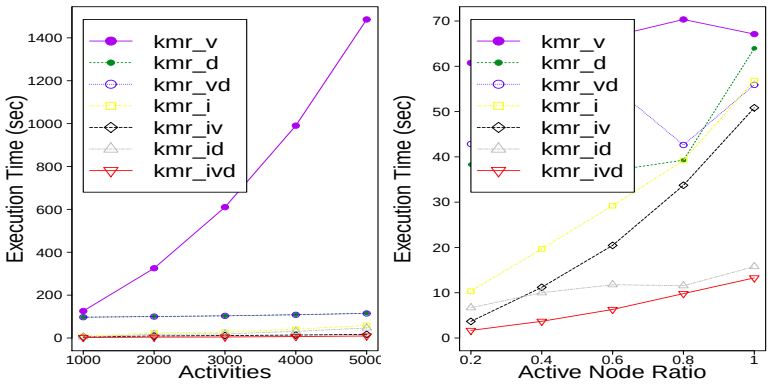
(a) Synthetic data (b) Real data

Figure 2.10: Scalability of KMR with increasing number of routes k

Effect of the Number of Routes

Again, both synthetic and real-world data sets were used to observe the effect of increasing the number of routes on execution time. For the synthetic data set, the number of nodes was set to $1,000$, the number of activities was $1,200$ and the active node ratio was 0.2. For the real-world data, the number of nodes was set to $15,000$, the number of activities for the $15,000$ nodes was 369, and the active node ratio for the $15,000$ nodes was 0.005. Figures 4.10 gives the execution times on the two data sets as the number of routes in the network increases. As before, KMR_{IVD} performs the best.

Computational savings increase when performance-tuning is introduced.



(a) Activities (b) Active Node Ratio

Figure 2.11: Scalability of KMR with increasing (a) number of activities and (b) active node ratio on synthetic data

Effect of the Number of Activities

Only synthetic data was used to observe the effect of activity number, since this parameter cannot be varied in real data. The number of nodes was set to $1,000$, the number of routes k was 2, and the active node ratio was 0.2. As shown in Figure 2.11(a), KMR_{IVD} performs the best followed by KMR_{IV} and KMR_{ID}. Computational savings increases when performance-tuning is introduced.

Effect of Active Node Ratio

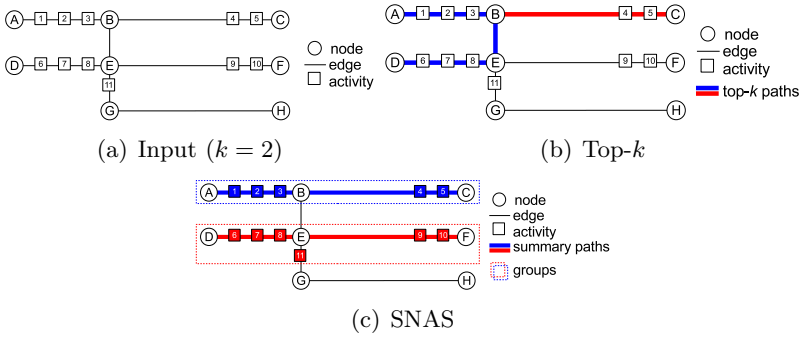
Recall that the active node ratio (ANR) is the number of active nodes divided by the total number of nodes. Here again, only synthetic data was tested because the parameter cannot be varied in the real data set. The number of nodes was set to $1,000$, the number of routes k was 2, and the number of activities was $1,200$. Figure 2.11(b) shows that KMR_{IVD} performs the best followed by KMR_{ID} and KMR_{IV}. Computational savings increases with higher active node ratios due to NOVA_TKDE, D-SPARE_TKDE, and inactive node pruning.

In summary, the experiments uniformly show that KMR_{IVD}, the combination of all three performance-tuning decisions, gives the best results, much better than KMR_I [11]

alone. This is because KMR_{IVD} filters shortest paths that are not between active nodes, enables faster assignment of activities, and enables faster summary path calculation.

2.7 Discussion

Individual runs of KMR are greedy like individual executions of the K-Means algorithm and risk convergence to local minima. For example, Table 2.4 shows four pairs of initial seeds in the sample network dataset shown in Figure 2.12 and the corresponding final solutions. When initialized with seeds of $\langle A, B\rangle$ and $\langle D, E\rangle$, KMR converges to a final solution of $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$ with a total activity coverage of 10 as shown in the first row of Table 2.4. However, when initialized with $\langle A, B, E, D\rangle$ and $\langle B, C\rangle$, the algorithm converges to initial seeds (a local minima) with a total coverage of 8. Just like K-Means, the solution quality of KMR, may be enhanced by running it multiple times with different initial seeds.



(a) Input ($k = 2$) (b) Top-k

(c) SNAS

Figure 2.12: Example (a) Input and Output of (b) Top-k queries and (c) SNAS (Best in color).

There is more than one way to detect linear activity concentration. Different formulations result in different outputs. For example, top-k queries return the k shortest paths whereas SNAS returns groups of activities where each group is represented by a summary path with maximum activity coverage. When initially seeded with the results of the top-k query, KMR's results will be no worse in terms of activity coverage as shown in bottom two rows of Table 2.4. However, KMR can find better solutions with different seeds as shown in first two rows of Table 2.4.

Table 2.4: Initial seeds and final solutions of KMR

Initial Seeds	Final Result	Activity Coverage
$\langle A, B\rangle$ and $\langle D, E\rangle$	$\langle A, B, C\rangle$ and $\langle D, E, F\rangle$	10
$\langle A, B\rangle$ and $\langle E, F\rangle$	$\langle A, B, C\rangle$ and $\langle D, E, F\rangle$	10
$\langle A, B, E, D\rangle$ and $\langle B, C\rangle$	$\langle A, B, E, D\rangle$ and $\langle B, C\rangle$	8
$\langle A, B, E, D\rangle$ and $\langle E, F\rangle$	$\langle A, B, E, D\rangle$ and $\langle E, F\rangle$	8

Figure 2.12 shows example outputs of top-k queries and SNAS on a given network for $k = 2$. In this case, the top-k paths are constrained to be disjoint shortest paths in terms of edges (versus overlapping paths) and the two paths with the most activities are $\langle A, B, E, D\rangle$ (Activity Coverage = 6) and either $\langle B, C\rangle$ (as illustrated) or $\langle E, F\rangle$ (Activity Coverage = 2), resulting in a total activity coverage of 8. Top-k queries are not designed to group activities and thus activities that are not on a top-k path do not belong to any group (e.g., activities 9, 10, and 11). SNAS, by contrast, returns groups of activities represented by summary paths $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$ shown by the dashed lines (Activity Coverage = 10). $\langle A, B, C\rangle$ and $\langle D, E, F\rangle$ are representatives that maximize activity coverage for the set of activities in each of their respective groups. As can be seen, the output of SNAS is different from that of top-k. An activity that is not on a summary path in SNAS may still be a member of the same group (e.g., activity 11 in Figure 2.12(c)) whereas the notion of groups does not exist in top-k queries.

2.8 Conclusion

This work explored the problem of spatial network activity summarization in relation to important application domains such as preventing pedestrian fatalities and crime analysis. We proposed a K-Main Routes (KMR) algorithm that discovers a set of k shortest paths to group activities. KMR uses inactive node pruning, Network Voronoi activity Assignment (NOVA_TKDE) and Divide and conquer Summary PAth REcomputation (D-SPARE_TKDE) to enhance its performance and scalability. We presented a case study comparing KMR with other summarization techniques on pedestrian fatality data. Experimental evaluation using both synthetic and real-world data sets indicated that the performance-tuning decisions utilized by KMR yielded substantial computational savings without reducing the coverage of the resulting summary paths.

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). We plan to investigate a distance-based rather than coverage-based objective function. We will also generalize SNAS for all paths and explore spatial constraints (e.g., nearest neighbors).

In an alternative formulation of the problem, activities could be modeled as simple edge weights. This might be useful for applications where summarization does not require an exact location for a given activity. We intend to investigate this in future work. We also plan to characterize graphs where D-SPARE_TKDE will find a path within a given fragment as well as extend our approach to account for different types of activities. Finally, incorporating time into activity summarization and choosing the right k will be explored.

Algorithm 2 AssignActivitiesToSummaryPaths

Input:

1) a spatial network $G = (N, E)$,
2) a set of activities A,
3) a number of routes k,
4) a set of summary paths $\hat{P}$,
5) mode $\in \{naive, NOVA\_TKDE\}$

Output:

k groups formed by assigning each activity $\in A$ to the closest summary path $\in \hat{P}$.

Algorithm:

if mode = "naive" **then**
    Enumerate all distances between each activity $a_i \in A$ and each summary path $p_i \in \hat{P}$
    $currentGroups \leftarrow$ assign each a_i to the closest p_i
else if mode = "NOVA_TKDE" **then**
    $V \leftarrow$ virtual node connected to all nodes of all $p_i \in \hat{P}$ with zero-weight edges
    $Open, T_{nodes} \leftarrow V$; $Closed, T_{activities} \leftarrow \emptyset$
    repeat
        $n \leftarrow$ closest node in $Open$ to any $p_i \in \hat{P}$; $Closed \leftarrow n$
        update T_{nodes} with $x_i.distance$ and $x_i.sp$ for each of n's neighbors $x_i \notin Closed$
        if $x_i \notin Open$ **then** $Open \leftarrow x_i$
        update $T_{activities}$ with $a_i.distance$ and $a_i.sp$ for each activity $a_i \in edge(n, x_i)$
        $currentGroups \leftarrow$ assign each a_i to the closest p_i based on $T_{activities}$
    until all active nodes $\in Closed$ or $|Open| = 0$
    if $\exists$ unassigned activities, a_j **then**
        $currentGroups \leftarrow$ assign each a_j to any p_i
return $currentGroups$

Algorithm 3 RecomputeSummaryPaths

Input:

1) a spatial network $G = (N, E)$,
2) a set of activities A,
3) a number of routes k,
4) a set of groups $currentGroups$,
5) mode $\in \{naive, D\text{-}SPARE\_TKDE\}$

Output:

k summary paths, $\hat{P}'$, that maximize activity coverage (AC) for each group $\in currentGroups$

Algorithm:

```
1: if mode = "naive" then
2:     for each c_i ∈ currentGroups do
3:         P ← shortest paths between active nodes of G
4:         maxPath ← path in P with Max AC based on
           c_i's activities
5:         P̂' ← maxPath
6: else if mode = "D-SPARE_TKDE" then
7:     for each c_i ∈ currentGroups do
8:         P' ← the set of shortest paths between the active
           nodes of c_i
9:         maxPath ← path in P' with Max AC based on
           c_i's activities
10:        if maxPath = ∅ then
11:            P ← shortest paths between active nodes
               of G
12:            maxPath ← path in P with Max AC based
               on c_i's activities
13:        P̂' ← maxPath
14: return P̂'
```

Chapter 3

Geo-referenced Time-series Summarization Using k-Full Trees

3.1 Introduction

Geo-referenced or geographic Time-series (GTs) allow us to observe the evolution in time of some phenomenon in a fixed location, i.e., the time-changing value of an observed property [51]. Examples of GTs include the CIA World Factbook data for each year from 1989 to 2011 [52] and West Nile Virus cases for each of the 50 states in the US from 1999 to 2002. The ability to summarize geo-referenced time-series has potentially important applications for understanding the spread of protests, diseases, crimes, fires, pollutants, etc. For example, the geopolitical implications of the Arab Spring protests have drawn global attention [3, 53] and summarization of data related to these events is of interest to political geographers, peace and conflict researchers, economists, etc. In epidemiology, geo-refereced time series summarization may be useful for understanding the spread of disease so that patterns of progression can be established [54].

We define the problem of geo-refereced time series summarization (GTS) as follows: given a set of regions with activity counts at each time instant (e.g., a listing of countries with number of protests or disease cases over time), a spatial neighbor relation, and a positive integer k, find k trees that maximize activity coverage and impose a partition on the GTS. Section 4.2 presents formal definitions of the basic concepts and problem statement of GTS.

Table 3.1: Summarization framework for various data genres

Data Genre	Group Definition (Partitioning Criteria)	Group Representation Choices	Statistic
Relational Table (a set of rows)	a partition of rows	Distinct values of attributes (e.g., age-group)	sum, count, mean, etc.
Spatial (Euclidean Space)	a partition of space	points, polygons, ellipses, line-strings	sum, count, mean, etc.
Spatial Graph (Neighbor Relationship)	a partition of a graph	node, path, tree, MST, heaviest full tree	sum, count, mean, etc.
Geo-referenced Time-series (Fixed Neighbor)	a partition of a geo-referenced time-series	ST-node, ST-path, ST-heaviest full tree	sum, count, mean, etc.

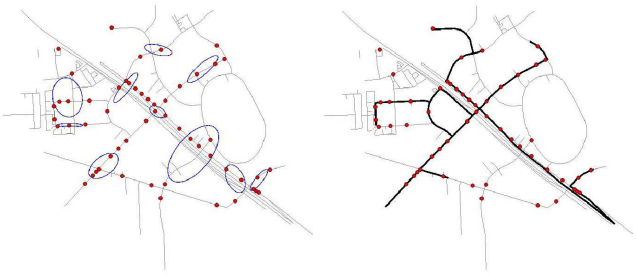
GTS is challenging for the following reasons. First, the computational complexity is high because of the large number of subsets of k trees due to the potential overlap between trees; there are an exponential number of choices when selecting k subsets from a set of trees. Second, a region with no activity may belong to a larger region with maximum activity coverage, making apriori-based pruning inapplicable. Determining appropriate interest measures for capturing event spread is important in capturing domain semantics. Examples of possible interest measures include maximum activity coverage and minimum average distance. Summarization based on maximizing activity coverage works well in domains such as epidemiology and geo-politics because it captures areas with the most activity (e.g., disease outbreak, protest). However, a region with maximum coverage might be composed of sub-regions with no activity which means that maximum activity coverage does not exhibit the anti-monotone property [55]. This makes pruning subsets based on maximum coverage more challenging since a region with no activity may be a part of a larger region with maximum activity coverage.

3.1.1 Data Summarization

Data summarization is an important concept in data mining that entails techniques for finding a compact description or representation of a dataset. The process typically involves defining a set of groups, finding a representative for each group, and reporting a statistic for each group (e.g., sum, mean, standard deviation). These notions differ depending on the genre of the data being summarized. Table 3.1 presents a summarization framework for four genres of data. Summarization methods listed for three of the genres already exist. An example of the first, relational table summarization, is the GROUP BY clause in SQL that is used to group rows having common values to report SQL aggregation functions such as mean and standard deviation. The group definition

in this case is a partition of rows and the group representation is distinct values of attributes such as age-groups, citizenship, income-group, etc.

Spatial Euclidean summarization includes heat maps and hotspot analysis. Heat maps provide a graphical representation of data in which individual values contained in a matrix are represented as colors. The group definition for heat maps might include a set of pixels and the group representation may be a subset of these pixels. Hotspots are a special kind of partitioned pattern where objects in hotspot regions have high similarity in comparison to one another and are dissimilar to all the objects outside the hotspot [9]. These spatial summarizations are based on spatial point locations where the group definition is a partition of space and the groups could be represented by points, polygons, ellipses, or line-strings. An example of a spatial summarization is given in Figure 3.1(a) where incidents of crime in a major US city are shown as dots and the spatial (Euclidean) summarization is represented using ten ellipses. The spatial summarization technique used in this case was K-Means [10].



(a) (b)

Figure 3.1: An example of (a) spatial summarization (ellipses) and (b) spatial graph summarization (paths).

Spatial graph summarization defines groups based on a partition of a graph and represents groups using nodes, paths, trees, etc. For example, the thicker lines in Figure 3.1(b) illustrate a summarization technique that uses linear representatives or paths to represent each partition [11]. Here ten paths are used to summarize the incidents of crime (dots) that are on the spatial graph (i.e., the road network).

Summarization of the last data genre listed in the table, geo-referenced time-series summarization, has been largely unexplored until now. It defines groups based on a

partition of a geo-referenced time series and may represent groups using spatio-temporal (ST)-nodes, ST-paths, or ST-heaviest full trees as is proposed in this work.

Finding a representative for each group and reporting a statistic for each group gives only a partial picture of the group. In many cases they give an effective summary of the group but not in all cases. This may lead to issues such as Simpson's paradox [56]. Thus one should examine groups (partitions) in addition to reviewing the statistics and group representatives for a complete story.

3.1.2 Related Work and their Limitations

Geo-referenced time-series summarization may be thought of as either anomaly-based or summarization-based. In anomaly-based approaches, anomalous hotspots or unusual regions are flagged. Anomaly-based approaches include the spatial and space-time (ST) scan statistic [57, 58, 59], emerging partition detection [60], irregular partition discovery [61], and STPC [62]. The spatial scan statistic uses a scanning window that varies its center and radius in order to scan the region under study when detecting spatial partitions. Emerging partition detection extends this idea to determine whether an observed increase in cases in a region is significant. Irregular partition discovery focuses on detecting partitions with irregular shapes where geographical boundary information is used to construct a graph in which a partition growing process is performed based on likelihood function maximization. STPC is a density-based technique for partitioning spatial and temporal polygons. Each polygon is required to have a minimum number of polygons in its neighborhood. Anomaly-based approaches do not summarize geo-referenced time-series. Instead they detect anomalous or unusual areas.

Summarization-based approaches, on the other hand, detect popular areas and aim to cover as many activities as possible.

3.1.3 Contributions

To the best of our knowledge, we are the first to propose techniques to summarize geo-referenced time-series. To address the challenges of the problem of GTS we propose the k-full tree (kFT) algorithm. In summary, our contributions are as follows:

- We formulate the problem of summarizing Geo-referenced Time Series (GTS)

using k-full trees

- We introduce a basic algorithm for the GTS problem. For reducing computational costs, we also introduce an algorithmic refinement called voronoi partition assignment (VPA) for partitioning regions.
- Experiments on real and synthetic datasets show that kFT with VPA leads to computational savings without affecting result quality.
- We present a case study on real data showing the output of our approach on Arab Spring data.

3.1.4 Scope and Outline

In this chapter, we use spatio-temporal full trees as representatives for partitions of GTS; alternate representations such as directed acyclic graphs or paths are not pursued in the present research. In a spatio-temporal full tree, every node other than the leaves has the same number of children and every leaf has the same depth. The trees are descriptive in nature and predictive notions such as causality are not explored.

The rest of this chapter is organized as follows: Section 4.2 presents the basic concepts and problem statement of GTS. Our proposed kFT algorithm and its algorithmic refinement are detailed in Section 3.3. Section 3.4 outlines the experimental evaluation. Section 4.5 presents a case study that shows the output of our approach on Arab Spring data. Section 3.6 concludes the chapter and previews future work.

3.2 Basic Concepts and Problem Statement

In this section, we introduce several key concepts in geo-referenced time-series summarization (GTS), and give a formal problem statement.

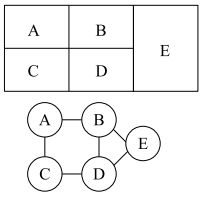
3.2.1 Basic Concepts

Relevant definitions to our problem statement and proposed approaches are as follows:

A **spatial framework, S**, is a partition of a region of geographic space, forming a finite tesselation of spatial objects [63]. In the plane, the elements of a spatial framework

will be polygons. A **spatial neighbor relation, SN** $\subseteq S \times S$ may be defined by the sharing of boundaries.

An example of a spatial framework are country boundaries and an example of spatial neighbors as defined by SN are neighboring countries such as the United States and Canada. For the spatial framework shown in Figure 3.2(a), polygons A and B are spatial neighbors while polygons C and E are not.



(a) A spatial framework

Polygon	Time Series
A	[5,5,0]
B	[2,5,0]
C	[0,5,0]
D	[1,0,5]
E	[0,5,1]

(b) A time-series attribute

Figure 3.2: A geo-referenced time-series.

A **temporal framework, T**, is a partition of a time interval into sub-intervals. A **time-series** is a computable function from T to a finite attribute domain, A_i. A **temporal neighbor relation, TN** $\subseteq T \times T$ and a time interval, t_i **immediately-follows** t_j iff $TN(t_i, t_j)$ and $t_i = t_j + 1$.

Months in a year is an example of a temporal framework where a calendar year is partitioned into months January, February, March, etc. January and February are temporal neighbors and February immediately follows January. Figure 3.2(b) shows another example of a temporal framework for each polygon, where $T \rightarrow Integers$. Polygon A has the time-series $[5, 5, 0]$ where each integer in the time-series may represent number of protests or disease cases.

A **Geo-referenced Time-series, GT** $= S \times T$. A **spatio-temporal node** in GT is denoted $\langle s_i, t_i \rangle$. A **spatio-temporal directed neighbor relationship, STN(**$\langle s_i,$ $t_i \rangle$**,** $\langle s_j,$ $t_j \rangle$**)** if and only if s_i and s_j are spatial neighbors and t_i immediately-follows t_j. s_i's value at t_i is the **activity count,** $c_{i,j}$, of s_i at t_i.

Figure 3.2 shows an example of a GT where the spatial framework comprising five regions (nodes), is shown on the left and the temporal framework for each polygon is shown on the right. Figure 3.3 shows a graph representation of Figure 3.2 with all the spatio-temporal directed neighbor relationships. Spatio-temporal nodes $A1$ and

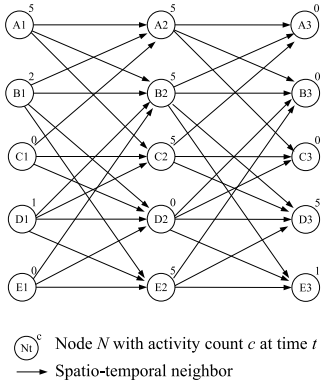


Figure 3.3: Spatio-temporal directed neighbor relationships.

$B2$ are spatio-temporal directed neighbors because they are spatial neighbors and t_2 immediately follows t_1.

A **spatio-temporal (ST)-full tree,** f is a tree of degree d and depth h such that

- $f \subseteq STN$
- non-leaves have degree d
- every leaf has the same depth h.

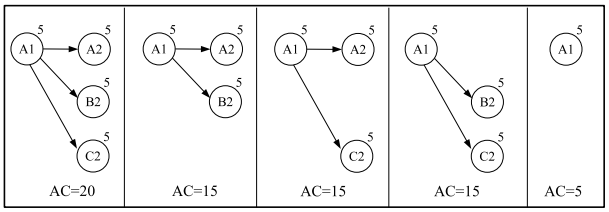


Figure 3.4: Examples of spatio-temporal (ST) full trees from Figure 3.3 rooted at $A1$.

Examples of spatio-temporal (ST) full trees are shown in Figure 3.4. $\langle A1, A2, B2, C2\rangle$ is a spatio-temporal full tree with degree 3.

A **Spatio-Temporal Field, STF** is a function from GT to a finite attribute domain, A_i.

Examples of STF include $GT \rightarrow Integers$, which denotes activity counts, such as number of disease cases and $GT \rightarrow Boolean$, which denotes event presence or absence

such as whether there were protests in a country at a given time during the Arab Spring revolutions.

The **activity coverage (AC)** of a ST-full tree, f, is the sum of activity counts, $\sum c_{i,j}$, for all spatio-temporal nodes $n \in f$.

In Figure 3.4, $AC(A1, A2, B2, C2)$ is 20 because the activity count for each of the four nodes in the full tree is 5, i.e., $AC(A1)+AC(A2)+AC(B2)+AC(C2) = 5+5+5+5 = 20$.

A **summary tree set** F is a collection of summary trees where each tree $f \in F$ is a full tree. A **summary tree** imposes a partitioning on a set of spatio-temporal nodes, N, such that $distance(n, f) \leq distance(n, f')\ \forall f' \in F,\ \forall n \in N$. The height of a summary tree is bounded by the number of time instants in N.

Figure 3.5(b) shows a summary tree $\langle A1, A2, B2, C2\rangle$ that imposes a partitioning on spatio-temporal nodes $A1$, $A2$, $A3$, $B1$, $B2$, $B3$, $C1$, $C2$, and $C3$.

3.2.2 Problem Formulation and Statement

The problem of geo-referenced time-series summarization (GTS) can be expressed as follows:

Given:

- A geo-referenced time-series
- A spatial neighbor relation, R
- A positive integer, k

Find:

- k trees
- A partitioning of spatio-temporal nodes

Objective:

- Maximize the activity coverage (AC) across k trees

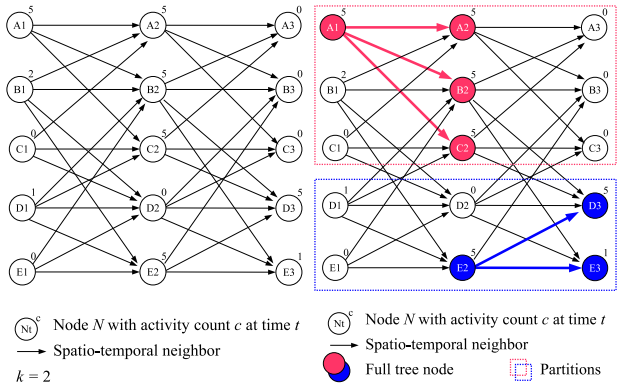
Constraints:

- Each tree is a spatio-temporal full tree and represents one source rooted at spatio-temporal node n

- Trees are mutually non-overlapping

The geo-referenced time-series input for GTS is defined in Section 3.2.1, R defines the spatial relationship between the regions of the geo-referenced time-series, and k represents the desired number of trees. The output for GTS is a set of k trees that maximize activity coverage and impose a partition on the GTS. The constraints are that the k trees are spatio-temporal full trees (Section 3.2.1), represent one source, and are mutually non-overlapping.

Example. Figure 3.5 shows input and output examples of GTS. For the input, the GTS and spatial neighbor relation from Figure 3.2 is used. Figure 3.5(a) adds all the spatio-temporal edges for each node and the number of trees k is set to 2. The output in Figure 3.5(b) shows 2 full trees that maximize activity coverage.



(a) GTS Input (b) GTS Output

Figure 3.5: Example (a) input and (b) output of GTS (Best in color).

3.3 Proposed Approach

This section describes the k-full tree (kFT) algorithm and our algorithmic refinement, voronoi partition assignment (VPA).

3.3.1 Basic k-Full Tree (kFT) Algorithm

Algorithm 4 presents the pseudocode for kFT, whose inputs are a geo-referenced time-series, a spatial neighbor relation, and a positive integer k, and whose outputs is a set

of k full trees that maximize activity coverage and impose a partition on the GTS. The basic structure of kFT resembles that of K-Means [10] in terms of selecting initial seeds, forming k partitions and updating the representative of each partition until the assignments no longer change. Line 1 of Algorithm 4 randomly selects k full trees as initial summary trees, which are the "seeds" for kFT. The algorithm then proceeds in two main phases. First, it forms k partitions by assigning each spatio-temporal node to its closest summary tree, i.e., the assignment step (line 3). Second, it calculates the summary tree of each partition that maximizes activity coverage, i.e., the update step (line 4). These two phases are repeated until the summary trees no longer change and the final summary trees and partitions are returned.

Algorithm 4 Basic k-Full Tree (kFT) Algorithm

Input:
1) A geo-referenced time-series,
2) A spatial neighbor relation, R,
3) A positive integer, k

Output:
k full trees that maximize activity coverage and impose a partition on the GTS

Algorithm:
1: Select k full trees as initial summary trees
2: **repeat**
3: **Phase 1:** Form k partitions by assigning each ST node to its closest summary tree
4: **Phase 2:** Recompute the summary tree of each partition
5: **until** summary trees do not change

Phase 1: Form k partitions by assigning each ST node to its closest summary tree

In phase 1, each spatio-temporal node, $n \in N$, is assigned to each of the given k summary trees, $f_i \in F$, to form k partitions. The assignment of n to a partition is based on the spatio-temporal distance from n to each spatio-temporal node u_i in f_i. The distance between n and f_i is the minimum distance between n and each $u_i \in f_i$. The distance between n and every f_i's is calculated and n is assigned to the closest summary tree. A basic method for phase 1 calculates a different shortest path between

each spatio-temporal node n and all k trees for all $n \in N$. Section 3.3.3 describes the voronoi partition assignment algorithm (VPA) that improves the runtime of phase 1.

Phase 2: Recompute the summary tree of each partition

Phase 2 computes and returns the summary tree with maximum activity coverage for each partition. All possible full trees that are rooted at each spatio-temporal node for each degree are calculated and the summary tree, f_{max}, with the highest activity coverage is stored. f_{max} is compared with the summary tree of the current partition and if it is found to have higher coverage, then the summary tree of the current partition is updated to f_{max}; otherwise the summary tree remains unchanged. A basic method for phase 2 enumerates all possible full trees within each partition and selects the one with maximum activity coverage.

3.3.2 Execution Trace

Figure 3.6 provides an execution trace of kFT. The input includes (1) a GTS with five nodes and three time instants (Figure 3.6(a)), (2) the spatial neighbor relationships shown in Figure 3.2, and (3) $k = 2$. kFT starts by selecting two trees as initial summary trees, namely $\langle A1 \rangle$ and $\langle E1 \rangle$ (Figure 3.6(b)). Next, two partitions are formed by assigning each spatio-temporal node to the nearest summary tree. The partitions formed are shown in Figure 3.6(c) and include nodes $A1, A2, A3, B1, B2, B3, C1, C2, C3$ being assigned to $\langle A1 \rangle$, and nodes $D1, D2, D3, E1, E2, E3$ being assigned to $\langle E1 \rangle$. For example, $A2$ is assigned to $\langle A1 \rangle$ since its distance to $\langle A1 \rangle$ is 1 and its distance to $\langle E1 \rangle$ is 3.

Within each partition, kFT selects the full tree with the maximum activity coverage to be the new summary tree (Figure 3.6(d)). If the current summary tree has coverage greater than or equal to the tree with maximum coverage, then the summary tree remains unchanged; otherwise, the summary tree is updated. In this case, summary tree $\langle A1 \rangle$ changes to $\langle A1, A2, B2, C3 \rangle$, and summary tree $\langle E1 \rangle$ changes to $\langle E2, E3, D3 \rangle$. Figure 3.4 shows the possible summary trees rooted at $A1$ and their respective activity coverages. Every possible summary tree for each spatio-temporal node is considered.

Phases 1 and 2 of kFT, which are depicted in Figures 3.6(c) and 3.6(d), are repeated until the summary trees do not change. In this example, phases 1 and 2 are repeated

once and since the summary trees do not change, the algorithm terminates, returning the partitions shown in Figure 3.6(d).

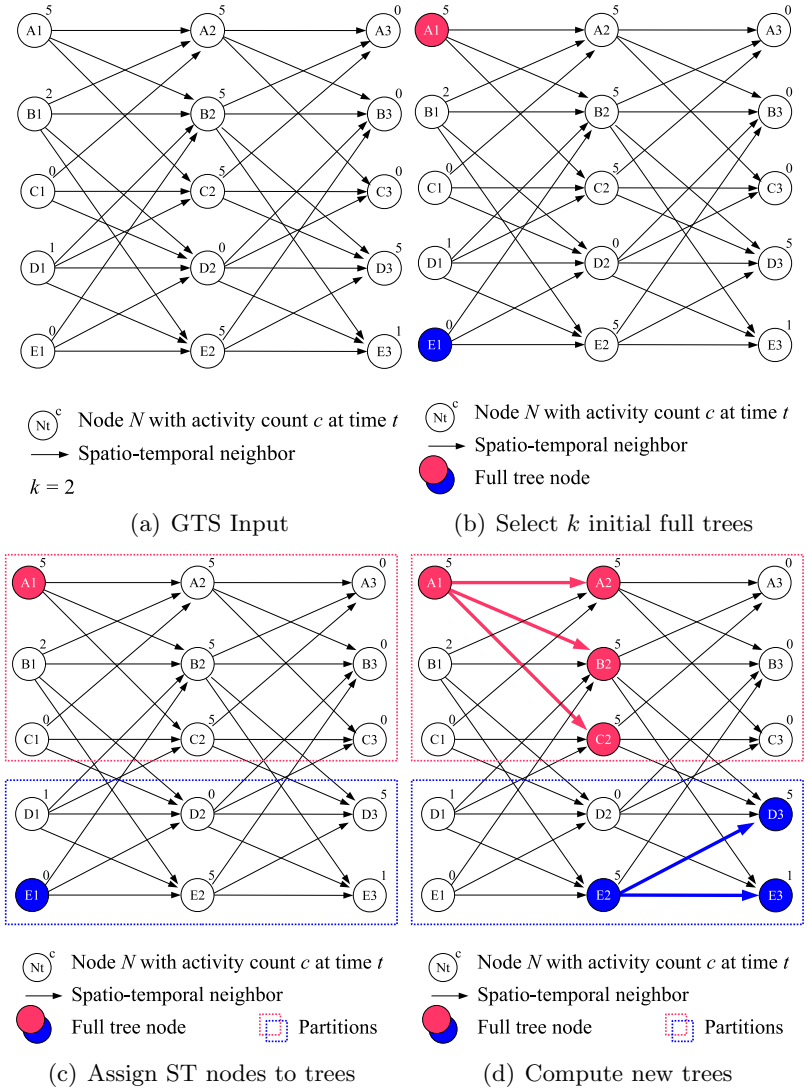


Figure 3.6: kFT Execution Trace (Best in color).

3.3.3 Refinement 1: Voronoi Partition Assignment

VPA aims to improve the runtime of phase 1, which forms k partitions by assigning ST nodes to full trees. The pseudocode for VPA is presented in Algorithm 5. The basic

idea of VPA is to perform one distance calculation between the nodes of the summary trees and every other spatio-temporal node versus performing multiple calculations as described earlier. Figure 3.7 shows an example of VPA. A virtual node V is connected to each node of the summary trees $\langle A1 \rangle$ and $\langle E1 \rangle$. $A2$ is assigned to $\langle A1 \rangle$ since $A1$ is a node on the shortest path from V to $A2$, i.e., $V, A1, A2$, whereas $E2$ is assigned to $\langle E1 \rangle$ since $E1$ is a node on the shortest path from V to $E2$, i.e., $V, E1, E2$.

Algorithm 5 Voronoi Partition Assignment

Input:

A set of k summary trees

Output:

A set of k partitions formed by assigning each spatio-temporal node, n, to one summary tree $f \in F$, where $distance(n, f) \leq distance(n, f')\ \forall f' \in F,\ \forall n \in N$.

Algorithm:

```
1:  Open ← all nodes of all f_i ∈ F
2:  Closed ← ∅
3:  T_nodes ← all nodes of all f_i ∈ F
4:  repeat
5:      n ← node in Open that is nearest to a summary tree
6:      Closed ← n
7:      for each x_i ∈ Adj[n] do
8:          if x_i ∉ Closed then
9:              update x_i.prev, x_i.distance, x_i.f in T_nodes
10:             assign x_i to the closest summary tree based
                on T_nodes
11:             if x_i ∉ Open then
12:                 Open ← x_i
13: until all nodes ∈ Closed
```

3.4 Experimental Evaluation

We experimentally evaluated kFT with and without the VPA algorithmic refinement on both synthetic and real-world data. For each workload experiment, we ran two versions of kFT:

- Basic kFT (kft)
- kFT with VPA (kft_v)

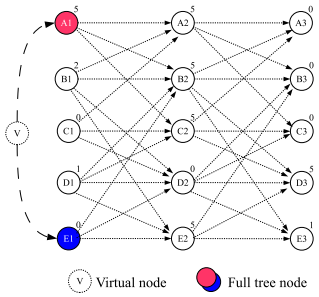


Figure 3.7: Voronoi Partition Assignment Example. $A2$ is assigned to $\langle A1 \rangle$ since $A1$ is a node on the shortest path from V to $A1$, i.e., $V, A1, A2$ (Best in color).

The goal of our experiments was a comparative analysis in which we investigated:

- How do candidate algorithms compare on computational cost? Subquestions included the following:
  - What is the effect of number of nodes?
  - What is the effect of number of time-intervals?
  - What is the effect of number of trees, k?

All experiments were performed on a Mac Pro with a 2 x Xeon Quad Core 2.26 GHz processor and 16 GB RAM.

3.4.1 Experiment Data Sets

Our experiments were performed on both synthetic and real-world data. To generate our synthetic geo-referenced time-series, we used the US Census Bureau's 2011 TIGER/Line Shapefiles for Hennepin County, Minnesota [50]; the nodes were the intersections of the road network. For each node, activity counts for each time instant were set to a number between 0 and 100.

The real-world data set we used was from the CIA World Factbook, which provides information on the history, people, government, economy, geography, communications, transportation, military, and transnational issues for 267 world entities [52]. This factbook is an excellent source of geo-referenced time-series because for each country we are given its spatial location and time-changing values of many quantitative variables of

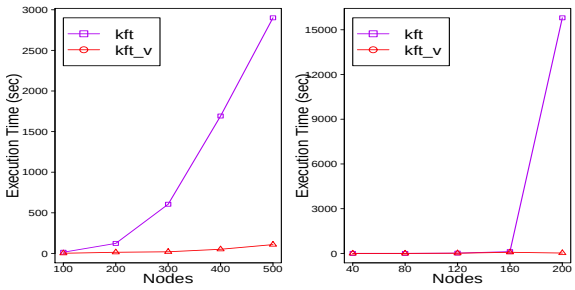
interest. For our experiments, we used the unemployment rate variable for each country. The spatial relationships for each country were derived using the sharing of boundaries.

Understanding the human terrain of culture, politics, and economics of a region is of global importance and datasets such as the CIA World Factbook contain a wealth of information that could be explored or summarized analytically. New methods to explore this data can potentially deepen our understanding of world events such as the Arab Spring [53].

3.4.2 Experimental Results

Our experimental results are as follows:

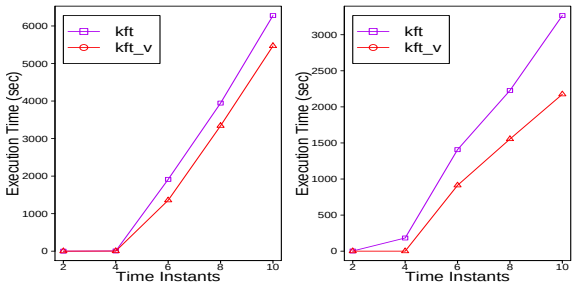
What is the effect of number of nodes? Both synthetic and real-world data sets were used to observe the effect of increasing the number of nodes on execution time. For the synthetic data, the number of time instants was set to 3 and k was set to 2. For the real-world data, the number of time instants was also set to 4 and k was also set to 2. Figure 4.10(a) gives the execution times for both data sets. As can be seen, computational savings increases as the number of nodes increases due to VPA.



(a) (b)

Figure 3.8: Effect of number of nodes on (a) synthetic and (b) real data

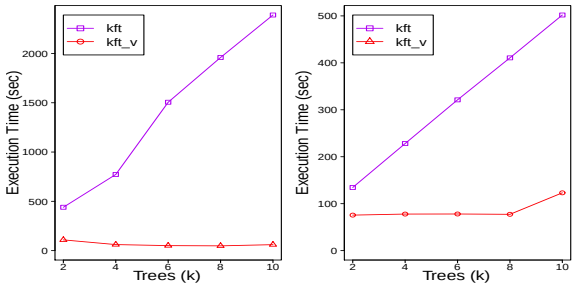
What is the effect of number of time instants? Again, both synthetic and real-world data sets were used to observe the effect of increasing the number of time instants on execution time. For the synthetic data, the number of nodes was set to 50 and k was set to 2. For the real-world data, the number of nodes was also set to 50 and k was also set to 2. Figure 3.9 gives the execution times for both data sets. As before, computational savings increase when the VPA algorithmic refinement is introduced.



(a) (b)

Figure 3.9: Effect of number of time instants on (a) synthetic and (b) real data

What is the effect of number of trees, k? The effect of increasing the number of trees on execution time was observed using both synthetic and real-world data sets. For the synthetic data, the number of nodes was set to 500 and the number of time instants was set to 3. For the real-world data, the number of nodes was set to 100 and the number of time instants was set to 3. Figure 3.10 gives the execution times for both data sets. As can be seen, computational savings increase when VPA is introduced.



(a) (b)

Figure 3.10: Effect of number of trees on (a) synthetic and (b) real data

Interpretation: The results of the series of experiments are uniformly that kft with voronoi partition assignment clearly gives the best results, much better than kft alone.

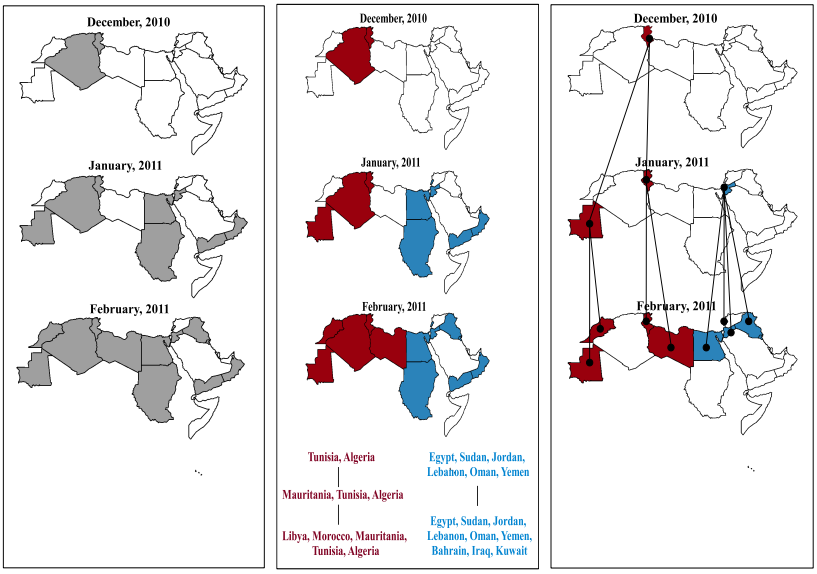
3.5 Case Study

We also conducted a qualitative evaluation of kFT that illustrates its output in applications involving geo-referenced time-series. Figure 4.9 presents the results of our case study on Arab Spring data [3].

The input is shown in Figure 3.11(a) and entails the geo-referenced time-series comprised of countries of the Arab League with boolean values for each country indicating whether there were protests from December 2010 to February 2011 [3]. The activity count for each country was set to 1 if protests occurred during the month (0 otherwise) and k was set to 2. December's data for each country is shown on top, January in the middle, and February on the bottom. The spatial relationships were derived using a distance threshold.

The output of kFT includes a partitioning of spatio-temporal nodes shown in Figure 3.11(b). The two groups or partitions that kFT produces are represented by different colors. For example, the maroon partition (partition 1) has members (Tunisia, Algeria) - (Mauritania, Tunisia, Algeria) - (Libya, Morocco, Mauritania, Tunisia, Algeria) and the blue partition has members (Egypt, Sudan, Jordan, Lebanon, Oman, Yemen) - (Egypt, Sudan, Jordan, Lebanon, Oman, Yemen, Bahrain, Iraq, Kuwait). The 2 trees or representatives for the partitions are shown in Figure 3.11(c), which is a sparser representation of the groups in Figure 3.11(b). For example, the maroon partition shows more members in Figure 3.11(b) but the representative in Figure 3.11(c) is a full tree. The kFT algorithm provides groups and group representatives of the actual protest data and the output shows how geo-referenced time-series may be summarized using this approach.

The lines in Figure 3.11(c) give examples of how the components are connected. As discussed, the goal of GTS is to summarize in a descriptive manner. GTS is not predictive and notions such as causality are not explored. Additionally, the temporal resolution of the data was a month. For example, although protests in Tunisia started on December 18, 2010 and protests in Algeria started on December 28, 2010, both are recorded as having protests in December 2010, with no distinction as to which started first based on the granularity of a month. Finer temporal resolutions (e.g., weekly) and accounting for activity frequencies may provide a better summarization and we plan to



(a) Actual Protests (b) Groups ($k = 2$) (c) Group Representation - k full trees ($k = 2$)

Figure 3.11: kFT output on Arab Spring data from December 2010 to February 2011 [3] (Best viewed in color).

evaluate this in the future. We also plan to get user feedback as to their preference of groups (Figure 3.11(b)) versus representatives (Figure 3.11(c)) and we will be evaluating kFT on other real data sets such as GDP in CIA world factbook data [52].

3.6 Conclusions

This work introduced the problem of geo-referenced time-series summarization (GTS) using k full trees. This problem is important for summarizing the spread of events over space and time such as outbreaks of disease or the Arab Spring uprisings. However, this problem is computationally challenging because there are a large number of subsets of k-full trees due to the potential overlap of trees. We proposed a k full tree (kFT) algorithm to solve GTS. kFT is novel because it finds popular areas (i.e., full trees) with maximum coverage rather than anomalous or unusual areas. kFT uses Voronoi Partition Assignment to speed up summary tree calculation without reducing activity coverage. kFT was validated using experimental evaluation on real and synthetic data

and a case study on Arab Spring data. Experiments demonstrate the computational savings of running kFT with VPA.

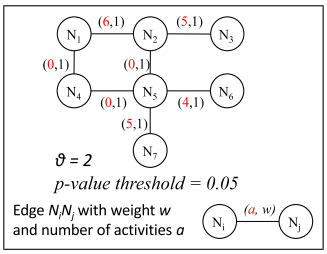
In future work, we plan to develop an algebraic cost model and investigate alternate interest measures (e.g., distance, density, etc). We plan to explore other partition representatives such as directed acyclic graphs. We also plan on refining our approach to attain more effective summaries by accounting for finer temporal resolutions and frequencies. Choosing the right k and determining kFT's usefulness to domain professionals will also be explored.

Chapter 4

Significant Linear Hotspot Discovery

4.1 Introduction

Significant Linear Hotspot Discovery (SLHD) identifies routes with significant concentrations of an activity, such as accidents or crimes. Informally, the SLHD problem can be defined as follows: given a spatial network, a collection of activities (e.g., pedestrian fatality reports, crime reports), and a likelihood threshold θ, find all shortest paths in the spatial network where the concentration of activities is unusually high (i.e., statistically significant) and the likelihood exceeds θ. Depending on the domain, an activity may be the location of a pedestrian fatality, a carjacking, a train accident, etc. Figures 4.1(a) and 4.1(b) illustrate an aggregated/simplified input and output example of SLHD, respectively. The input consists of seven nodes, seven edges (with edge weights set to 1 for illustration purposes, shown as the second number on each edge), twenty activities (shown as the first number in red on each edge), and $\theta = 2$, indicating that we are interested in shortest paths whose likelihood exceeds $\theta = 2$. The output contains two shortest paths, $\langle N_1, N_2, N_3 \rangle$ and $\langle N_6, N_5, N_7 \rangle$ that are at least twice as likely to have instances of the activity (e.g., accidents, crime).



(a) Input

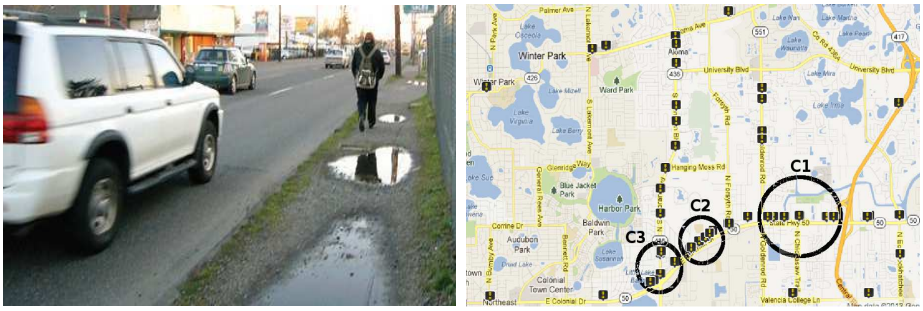
Path	λ	P-value
<N1,N2,N3>	3.05	0.007
<N6,N5,N7>	2.04	0.026

(b) Output

Figure 4.1: Simplified example of Significant Linear Hotspot Discovery where activity information is aggregated by edge (Best in color).

4.1.1 An Illustrative Application Domain: Preventing Pedestrian Fatalities

To illustrate the applicability of SLHD, 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 and 2009 [2]. 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 [2].



(a) (b)

Figure 4.2: (a) Pedestrian at risk on a road without proper sidewalks [2] (b) Pedestrian fatalities occurring on arterials in Orlando, FL [1]. A large fraction of the bounding circles (e.g., C1, C2) of significant routes are empty (Best in color).

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 [2]. Daily activities have shifted away from main streets towards higher speed arterials based on the emphasis on traffic movement. 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 4.2(a)). They lack sidewalks, crosswalks (or have crosswalks spaced too far apart), pedestrian refuges, street lighting, and school and public bus shelters [2].

This lack of basic infrastructure can be lethal. For example, forty percent of fatalities occurred where no crosswalk was available [2]. Figure 4.2(b) shows a map of pedestrian fatalities that occurred on Orlando roads from 2000 - 2009. Transportation planners and engineers need tools to assist them in identifying which frequently used road segments/stretches pose statistically significant levels of risks for pedestrians and consequently should be redesigned. 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.

4.1.2 Challenges

SLHD is challenging due to the potentially large number of candidate routes ($\sim 10^{16}$) in a given dataset with millions of activities or road network nodes. For large roadmaps such as the 100 million road-segments in the US, this results in prohibitive shortest path computation times. Additionally, significance testing does not obey the monotonicity property, meaning that there is no ordering between the likelihood of a path and its super-paths, or vice-versa. In other metrics such as activity count, for example, a path will always have less than or equal to the number of activities of its super-paths, a property which may be exploited for computational speedup. However, this property does not hold for significance testing. 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).

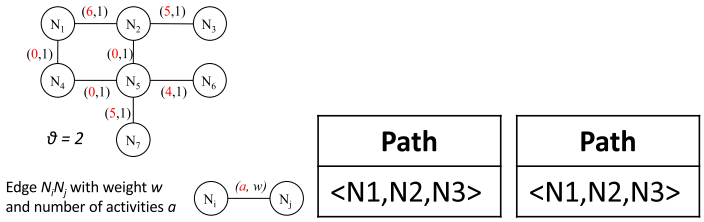
4.1.3 Related Work and Their Limitations

Dividing spatial data into statistically significant groups is an important task in many domains (e.g., transportation planning, public health, epidemiology, climate science, etc.) [9]. Methods for this type of partitioning may generally be considered to be geometry-based or network-based.

Geometry-based techniques [57, 64, 65] partition spatial data using two-dimensional 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 [57]. The spatial scan statistic employs a likelihood ratio test where the null hypothesis is the probability that disease inside a region is the same as outside, and the alternate hypothesis is that there is a higher probability of disease inside than outside. All the spatial regions, represented by a circle or rectangle 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, as shown in Figure 4.2(b). 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.).

Network-based techniques [5, 66, 4, 67], on the other hand, leverage the underlying spatial network when partitioning spatial data. For example, Linear Intersecting Paths (LIP) [4] and Constrained Minimum Spanning Trees (CMST) [5] utilize a subgraph (e.g., a path or tree) to discover statistically significant groups.

In LIP [4], one anomalous sub-component out of a set of connected paths that intersect each other is discovered. The connected paths are based on locations in the spatial network with the highest percentage of activities, specified by the user. Hence the likelihood ratio is only evaluated on a portion of the graph specified by this percentage, not the entire spatial network. Figure 4.3 shows an example input and output of LIP. The user-specified percentage is 30%, which means all the candidates will have



(a) Input (b) LIP Output (c) CMST Output

Figure 4.3: Simplified example (a) Input, where activities are summarized by total count per edge (b) Output of Linear Intersecting Paths (LIP) [4], and (c) Output of Constrained Minimum Spanning Trees (CMST) [5] (Best in color).

paths containing edge $\langle N_1, N_2 \rangle$ since this edge has six activities (out of a possible 20 activities). Examples of possible candidates are $\langle N_1, N_2, N_3 \rangle$, $\langle N_1, N_2, N_5 \rangle$, $\langle N_2, N_1, N_4 \rangle$, $\langle N_1, N_2, N_5, N_7 \rangle$, etc. The output is $\langle N_1, N_2, N_3 \rangle$, since it has the highest likelihood (Section 4.2 details how the likelihood ratio is calculated). 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 [5], finds one statistically significant tree in the spatial network. Figure 4.3(c) shows an example of this approach. Here the output is $\langle N_1, N_2, N_3 \rangle$, since this tree has the highest likelihood. However, 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.

Previous network-based techniques also face a fundamental limitation in dealing with very long edges with a dense cluster of activities on one end (e.g., edge $\langle N_1, N_2 \rangle$ in Figure 4.4). Long empty portions of these edges reduce the likelihood ratio of the entire edge and such edges are often excluded by previous approaches even though many users may prefer to include the portions with dense clusters of activities. In other words, previous approaches lack the capability to consider paths between activity locations and instead only focus on paths between network nodes. Consequently, they may miss

paths containing activities that are close to each other but are on a long edge.

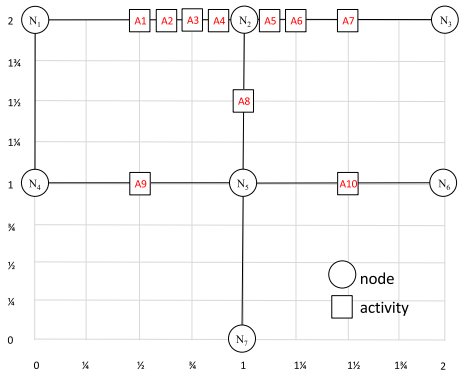


Figure 4.4: Example of dynamic segmentation where paths between activities at the sub-edge level are considered.

To address this limitation, this chapter investigates *dynamic segmentation* (paths between activities at the sub-edge level are considered) and we elaborate on it in the proposed approach section. Figure 4.4 shows an example of a spatial network that has been dynamically segmented. Dynamic segmentation facilitates the inclusion of paths with dense portions of activities such as $\langle A1, A2, A3, A4, A5, A6, A7\rangle$, which has a higher likelihood ratio (14) than that of the enclosing path $\langle N_1, N_2, N_3\rangle$ (5.83) in the traditional network.

Previously, we proposed the Significant Route Miner using Likelihood Pruning and Monte Carlo speedup as algorithmic refinements for computational speedup (SRM_GIS) [68]. We experimentally evaluated SRM_GIS and presented a case study comparing its output with SaTScan [69]. However, our previous approach was limited because it did not consider dynamic segmentation.

4.1.4 Contributions

In this chapter, we present a new dynamic segmentation model to improve the results of significant linear hotspot discovery. To the best of our knowledge, the proposed approach is the first to consider dynamic segmentation when discovering multiple statistically significant routes in the spatial network. We present new algorithmic refinements for dynamic segmentation. We also present a cost model for the proposed techniques and

prove that our proposed algorithmic refinements are correct. Specifically, our research contributions are as follows:

- We introduce a dynamic segmentation model for significant linear hotspot discovery.
- We propose new algorithms featuring 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 algorithmic refinements to dynamic segmentation, including a hierarchical filter and an active node filter.
- We analytically prove the correctness of the proposed algorithmic refinements and present a cost analysis.
- We present a case study comparing the proposed significant network-based analysis (i.e., shortest paths) to a significant geometry-based analysis (e.g., circles) on pedestrian fatality data.
- Experimental results on real and synthetic data show that the proposed algorithm, with our refinements, yields substantial computational savings over a naïve approach without reducing result quality.

4.1.5 Scope and Outline of the Chapter

This work focuses on finding significant discrete activity events (e.g., pedestrian fatalities, crime reports) 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 [38] 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. A dynamically changing number of activities is presently beyond the scope of this research, as are techniques that do not explore statistical significance (e.g., DBScan [6], K-Means [10], KMR [11], and Maximum Subgraph Finding [70]). Modeling stochastic route choice (where one or several of the edge attributes are not deterministic [71]) and the factors underlying the identified significant linear hotspots also falls outside the scope of this chapter.

The chapter is organized as follows: Section 4.2 presents the basic concepts and problem statement of SLHD. Section 4.3 presents our preliminary results towards addressing SLHD. Section 4.4 details our proposed dynamic segmentation approaches. Section 4.5 presents a case study comparing the proposed significant network-based output (i.e., shortest paths) to a significant geometry-based output (e.g., circles) on pedestrian fatality data. The experimental evaluation is covered in Section 4.6. Section 4.7 presents a discussion outlining potential future directions and Section 4.8 concludes the chapter.

4.2 Basic Concepts and Problem Statement

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

4.2.1 Basic Concepts

We define our basic concepts as follows:

Definition 8. *A **spatial network** $G = (N, E)$ consists of a node set N and an edge set E, where each element u in N is associated with a pair of real numbers (x, y) representing the spatial location of the node in a Euclidean plane [40]. Edge set E is a subset of the cross product $N \times N$. Each element $e = (u, v)$ in E is an edge that joins node u to node v.*

Figure 4.1(a) 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 9. *AR is a **set of activity reports**, where each activity report (or **activity**) ar_i has a location on an edge $e = (u, v)$. $a(e = (u, v))$ is the number of activities on an edge $e = (u, v)$.*

For simplicity we will use aggregated input (i.e., total activity on each edge) in the earlier part of this chapter. 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.

Each edge in Figure 4.1(a) is associated with a number of activities (e.g., edge $\langle N_1, N_2 \rangle$ has 6 activities).

Definition 10. *The **activity coverage inside a path**, a_p, is the number of activities on p. The **activity coverage outside** p is $|AR| - a_p$, where $|AR|$ is the total number of activities in the spatial network, G.*

For example, in Figure 4.1(a), the activity coverage *inside* path $\langle N_1, N_2, N_3 \rangle$ is 11 whereas the activity coverage *outside* $\langle N_1, N_2, N_3 \rangle$ is $20 - 11 = 9$.

Definition 11. *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 Figure 4.1(a), the weight *inside* $\langle N_1, N_2, N_3 \rangle$ is 2 whereas the weight *outside* $\langle N_1, N_2, N_3 \rangle$ is $7 - 2 = 5$.

Definition 12. *The **likelihood ratio of path** p, $\lambda_p = \frac{a_p \div w_p}{(|AR| - a_p) \div (|W| - w_p)}$ [67, 57].*

The likelihood ratio of path p, λ_p is the ratio of the activity density *inside* path p to the activity density *outside* p. Activity density may be estimated in different ways across different domains. In transportation planning, activity density inside p may be estimated using $\frac{a_p}{VMT}$, where VMT is vehicle miles traveled (i.e., the total number of miles driven by all vehicles within a given time period and geographic area). Path weight may also be used to estimate activity density [67]. In Figure 4.1(a), $\lambda_{\langle N_1, N_2, N_3 \rangle} = \frac{11 \div 2}{9 \div 5} = 3.05$.

Definition 13. *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 4.1(a), $\langle N_1, N_2, N_5, N_6 \rangle$ 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 $\langle N_1, N_2, N_5, N_6 \rangle$.

4.2.2 Problem Statement

The problem of Significant Linear Hotspot Discovery (SLHD) can be expressed as follows:

Given:

1. A spatial network $G = (N, E)$ with a set of 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 likelihood ratio (λ) threshold, θ,
3. A p-value,
4. m, indicating the number of Monte Carlo simulations

Find: All routes $r \in R$ with likelihood ratio $\lambda_r \geq \theta$ and a p-value significance level
Objective: Computational efficiency
Constraints:

1. Each route $r \in R$ is a shortest path between its end-nodes,
2. $r_i \in R$ is not a subset of any $r_j \in R \; \forall r_i, r_j \in R$ where $r_i \neq r_j$,
3. Correctness and completeness

The spatial network input for SLHD is defined in Definition 8. The θ input is a threshold indicating the minimum desired likelihood ratio. The p-value input is the desired level of statistical significance and m indicates the number of Monte Carlo simulations for determining statistical significance. The output for SLHD is a set of shortest paths 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.

Example. The network in Figure 4.1(a) can be viewed as a road network, composed of streets (edges) and intersections (nodes). The aim is to find significant shortest paths that meet the given likelihood threshold of 2. In other words, find shortest paths that are twice as likely to have pedestrian fatalities. In a transportation planning scenario, identifying such routes would guide street redesign efforts to reduce the risk of pedestrian fatalities (e.g., adding sidewalks, crosswalks, pedestrian refuges, street lighting, etc.). In Figure 4.1(b), routes $\langle N_1, N_2, N_3 \rangle$ and $\langle N_6, N_5, N_7 \rangle$ are returned since they are shortest paths whose likelihood exceeds $\theta = 2$ and they are not sub-paths of any other path in the output.

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 truly anomalous. Here the null hypothesis states that the paths identified by the path likelihood ratio are random or by chance alone. The likelihood ratio is supplemented 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 value of a given likelihood ratio as equally or more extreme than that observed by chance alone.

In the Monte Carlo simulations, each activity in the original graph G is randomly associated with an edge 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; the original activities in G are now shuffled so they may be on different edges in G_s. We then compare the highest likelihood threshold λ_{maxG_s} of randomized G_s with the highest λ_{maxG} of original G. If the original one is smaller (i.e., $\lambda_{maxG} < \lambda_{maxG_s}$), then $p = p + 1$. The above process repeats m times and after it terminates, the p-value is subsequently p/m. Paths whose p-values are less than or equal to the given p-value threshold are deemed statistically significant.

4.3 Preliminary Results

We initially solved the SLHD problem with SRM_GIS [68] featuring two algorithmic refinements: Likelihood Pruning and Monte Carlo Speedup.

4.3.1 Naïve Significant Route Miner (NaïveSRM)

Algorithm 6 presents the pseudocode for the NaïveSRM approach. The basic idea behind the algorithm is to find all statistically significant shortest paths in the spatial network whose likelihood exceeds θ, under the constraint that the shortest paths returned are not sub-paths of any other path in the output. Algorithm 6 proceeds by calculating all shortest paths, P, in the spatial network (Line 1). Line 2 evaluates each shortest path in P to determine if it meets the given likelihood threshold θ to form a *Candidates* set. In line 3, 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 4, 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 6 Naïve Significant Route Miner (NaïveSRM) Algorithm

```
Input:
   1) A spatial network G = (N, E) with a set of activities with point locations on network nodes
   or edges and weight function w(u, v) > 0 for each edge e = (u, v) ∈ E (e.g., network distance),
   2) A likelihood ratio (λ) threshold, θ,
   3) A p-value threshold,
   4) m, indicating the number of Monte Carlo simulations
Output:
   All routes r ∈ R with λ_r ≥ θ and p-value significance level
Algorithm:
1: {Step 1:} P ← calculate all-pairs shortest path in G
2: {Step 2:} Candidates ← paths in P having λ ≥ θ
3: {Step 3:} SigRoutes ← significant paths in Candidates using m Monte Carlo simulations
4: {Step 4:} return paths that are not sub-paths of any other path in SigRoutes
```

NaïveSRM Example: Figure 4.5 shows an example execution trace of NaïveSRM. The spatial network has 7 nodes, 7 edges, and 20 activities, represented by the first number in red on each edge (e.g., edge $\langle N_1, N_2 \rangle$ has six activities). The given likelihood ratio threshold θ is set to 2 and the p-value is set to 0.05.

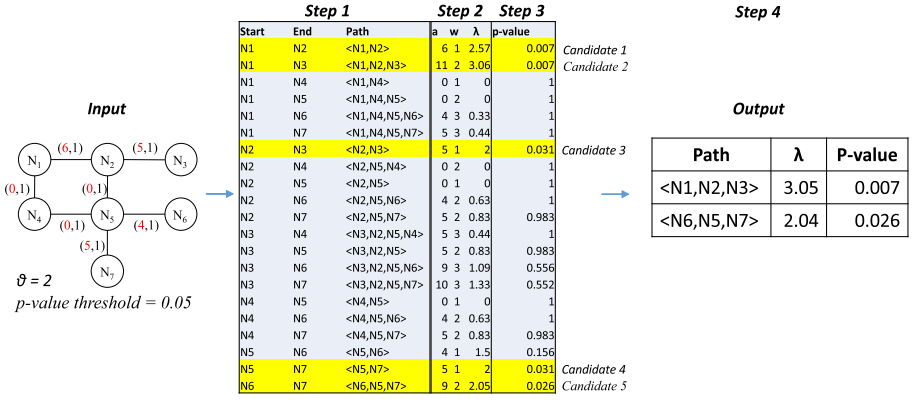


Figure 4.5: Execution trace of Naïve Significant Route Miner (NaïveSRM). Circles represent nodes and lines represent edges (Best in color).

In step 1 of Figure 4.5, all shortest paths in the given spatial network are calculated.

For example, the shortest path between nodes N_1 and N_3 is $\langle N_1, N_2, N_3 \rangle$. Next, in step 2, the likelihood ratio, λ, for each shortest path is determined (see Definition 12) and those whose $\lambda \geq \theta$ are stored as candidate solutions. In the figure, the five highlighted paths $\langle N_1, N_2 \rangle$, $\langle N_1, N_2, N_3 \rangle$, $\langle N_2, N_3 \rangle$, $\langle N_5, N_7 \rangle$, and $\langle N_6, N_5, N_7 \rangle$ are all candidates since their likelihood ratios meet or exceed the threshold of 2. In step 3, the statistical significance of each candidate is calculated using Monte Carlo simulations (discussed next). All five candidates meet the p-value threshold of 0.05. In step 4, the paths among significant paths that are not sub-paths of any other path are returned. In this example, paths $\langle N_1, N_2, N_3 \rangle$ and $\langle N_6, N_5, N_7 \rangle$ are returned. Paths $\langle N_1, N_2 \rangle$, $\langle N_2, N_3 \rangle$, and $\langle N_5, N_7 \rangle$ were not returned (even though they met and exceeded the likelihood and p-value thresholds) because they are each sub-paths of the two paths that were returned.

4.3.2 Significant Route Miner with Likelihood Pruning and Monte Carlo Speedup (SRM_GIS)

Algorithm 7 presents the pseudocode for our previous SRM_GIS [68] approach. SRM_GIS uses filter and refine techniques (e.g., Likelihood Ratio pruning and Monte Carlo speedup) to achieve computational savings. Filter and refine techniques may not change worst case complexity but they can reduce runtime in many cases. Likelihood Ratio pruning creates a boundary via the upperbound likelihood 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 likelihood 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 likelihood ratio in the simulated dataset beats the maximum likelihood ratio in the original dataset.

Likelihood Pruning

Likelihood pruning aims to avoid calculating all shortest paths in G based on the given threshold θ. It is based on the idea that for each shortest path p, it is possible to determine an upper bound likelihood ratio for the super-paths rooted at p's start node,

without calculating those super-paths.

Definition 14. *The **upperbound likelihood ratio for path** p,* $\hat{\lambda_p} = \frac{\hat{a_p} \div \hat{w_p}}{(|AR| - \hat{a_p}) \div (|W| - \hat{w_p})}$, *where* $\hat{a_p} = a_p + (|AR| - a_t)$ *(where* a_t *is the number of activities in the shortest path tree rooted at* p*'s source node) and* $\hat{w_p}$ *is the weight of the shortest super-path of* p*, rooted at* p*'s start node.*

The intuition behind the upper bound likelihood 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.

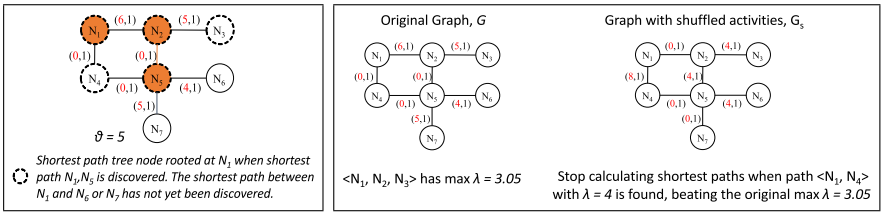
Algorithm 7 Significant Route Miner with Likelihood Pruning and Monte Carlo Speedup (SRM_GIS) Algorithm

```
Inputs and Outputs for SRM_GIS are same as NaïveSRM
Algorithm:
   {Step 1:  Likelihood Pruning}
1: for each s ∈ active nodes in G do
2:    Initialize D[v] ← inf; Pred[v] ← ∅; Λ̂[v] ← θ; a[v] ← 0; a_t ← 0; D[s] ← 0; PQ ← N
3:    while PQ ≠ ∅ do
4:       u ← node in PQ with smallest distance in D[]; P ← shortest path (s,u) in Pred[]
5:       a_t ← a_t + number of activities on edge Pred[u]
6:       if Λ̂[v] ≥ θ then
7:          for each v adjacent to u do
8:             sum ← D[u] + w(u,v)
9:             if sum < D[v] then
10:               D[v] ← sum; update v's position in PQ based on sum; Pred[v] ← u
11:               a[v] ← a[u] + a(u,v); ŵ ← sum+ weight of closest neighbor w(u,v)
12:               Λ̂[v] ← calculate λ̂_sv based on a[v], a_t and ŵ
13: {Step 2:} Candidates ← paths in P having λ ≥ θ
    {Step 3:  Monte Carlo Speedup}
14: λ_maxG ← highest likelihood ratio in G
15: for each simulation_1....simulation_m do
16:    G_s ← assign activities in G to random edges
17:    λ_maxG_si ← 0
18:    for each shortest path p ∈ G_s do
19:       if λ_p > λ_maxG_si then
20:          λ_maxG_si ← λ_p; p_maxr ← p_maxr + 1
21:          if p_maxr/N ≤ p-value threshold then return ∅
22:          if λ_p > λ_maxG then break
23:    for each route r ∈ Candidates do
24:       if λ_maxG_si > λ_maxG then p_r ← p_r + 1
25: for each route r ∈ Candidates do
26:    if p_r/N ≤ p-value threshold then SigRoutes ← r
27: {Step 4:} return paths that are not sub-paths of any other path in SigRoutes
```

Lines 1-12 of Algorithm 7 shows the pseudocode for likelihood pruning, which is

similar to Dijkstra's algorithm [41] with a few exceptions: (1) the shortest paths from a single active node to all destinations are calculated for all active nodes in the spatial network, (2) if the upper bound likelihood ratio for path $\langle s...u \rangle$ is below the given likelihood threshold θ, u's neighbors are not visited (line 6), and (3) upperbound statistics are calculated and updated each time the weight from source s to a node v is updated (lines 9-12).

Likelihood Pruning Example: Figure 4.6(a) illustrates the basic idea behind likelihood pruning. In this example, we have set the likelihood threshold to $\theta = 5$, indicating that we are interested in paths that are five times as likely to have pedestrian fatalities. During the algorithm's execution, at some point the source node becomes N_1, and the shortest path between N_1 and every other active node in the spatial network is calculated. When the shortest path between N_1 and N_5 is calculated, the upper bound likelihood ratio for path $\langle N_1, N_2, N_5 \rangle$ is determined to be 4, since based on Definition 14, the calculation would be $\frac{(6+(20-11))\div 3}{(20-((6+(20-11))\div(7-3)}$, where $\hat{a_p} = 6 + (20 - 11) = 15$ and $\hat{w_p} = 2+1 = 3$. We can, therefore, avoid calculating the shortest paths $\langle N_1, N_2, N_5, N_6 \rangle$ and $\langle N_1, N_2, N_5, N_7 \rangle$ for $\theta = 5$.



(a) (b)

Figure 4.6: (a) Example of Likelihood Pruning. Since we know the upper-bound likelihood for $\langle N_1, N_2, N_5 \rangle$ is 4, we can avoid calculating the shortest paths $\langle N_1, N_2, N_5, N_6 \rangle$ and $\langle N_1, N_2, N_5, N_7 \rangle$ for $\theta = 5$. (b) Example of Monte Carlo Speedup. (Best in color).

Lemma 6. *$\hat{a_p} = a_p + (|AR| - a_t)$ is a correct upper bound for the number of activities of all super-paths of p, rooted at p's source node.*

Proof. Consider the most current shortest path tree, t, generated when the shortest path p from node u to node v is found in the spatial network. By definition, p is unique in t. Thus, all super-paths of p, rooted at u will not contain any nodes or edges in t since t was generated when v was discovered (closed). Therefore, super-paths of p rooted at

u cannot contain any activities already in t, except those activities already in p. Hence, $\hat{a_p} = a_p + (|AR| - a_t)$ is a correct upper bound for the number of activities of p, rooted at p's source node. □

Lemma 7. $\hat{\lambda_p} = \frac{\hat{a_p} \div \hat{w_p}}{(|AR| - \hat{a_p}) \div (|W| - \hat{w_p})}$ *is a correct upper bound on the likelihood ratio for any super-path of* p*, rooted at* p*'s source node, where* a_t *is the number of activities in the shortest path tree rooted at* p*'s source node and* $\hat{w_p}$ *is the weight of the shortest super-path of* p*, rooted at* p*'s start node.*

Proof. $\hat{a_p} = a_p + (|AR| - a_t)$ is a correct upper bound for the number of activities of all super-paths of p, rooted at p's source node (Lemma 6). Since $\hat{w_p}$ is the weight of the shortest super-path of p rooted at p's source node, $\hat{\lambda_p}$'s value is maximized (based on its location in the denominators in Definition 14). □

Theorem 5. *Likelihood pruning is a correct pruning method*

Proof. A pruning method is considered correct if it does not exclude any optimal solutions (i.e., shortest paths whose likelihood ratio exceeds θ). By Lemma 7, $\hat{\lambda_p}$ is a correct upper bound on the likelihood ratio for any super-path of p rooted at p's source node. Since likelihood pruning does not prune out any path that meets or exceeds the upperbound likelihood ratio, and hence the likelihood ratio, no such optimal solutions will be pruned. Thus, likelihood pruning is correct. □

Monte Carlo Speedup

Monte Carlo speedup aims to calculate the p-value without considering 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 likelihood ratio than the maximum likelihood 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 likelihood ratio in the original graph. Additionally, Monte Carlo speedup stops all simulations the moment p out of m simulations are found where the simulated likelihood ratio beats the original maximum likelihood 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. The pseudocode for Monte Carlo speedup is presented in Lines 14-26 of Algorithm 7.

Monte Carlo Speedup Example: Figure 4.6(b) illustrates one of the basic ideas behind Monte Carlo speedup. In this example, the graph on the left is the original graph G whereas the graph on the right, G_s, represents one simulation with the activities shuffled. In G_s, instead of looking at all 42 shortest paths, we can stop and increment p the moment a path that has a likelihood higher than the maximum likelihood in G is found. In this case, that path would be $\langle N_1, N_4 \rangle$ (on the right of the figure), with a likelihood ratio of 4.

Theorem 6. *Monte Carlo Speedup is a correct method for calculating p-value.*

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

4.4 Proposed Approach

Recall the issue of dealing with very long edges with a dense cluster of activities on one end (e.g., edge $\langle N_1, N_2 \rangle$ in Figure 4.4). In this section, we present new proposed refinements to our SRM_GIS miner [68] to address this. Our refinements are based on the core idea of dynamic segmentation, and we present additional refinements to scale up our dynamic segmentation-based approach.

4.4.1 Dynamic Segmentation

Resolving statistically significant routes to the sub-edge level requires a dynamic segmentation data model. In our dynamic segmentation model, the network structure is altered such that new nodes are formed at the locations of activities and new edges are added to connect these nodes. Dynamic segmentation enables us to evaluate paths that start and end with activities, which may be in the middle of an edge in the original spatial network. As such, segments which were previously not tested for statistical

significance or which may have been previously deemed "not significant" because they were on a long edge, may end up as part of the result.

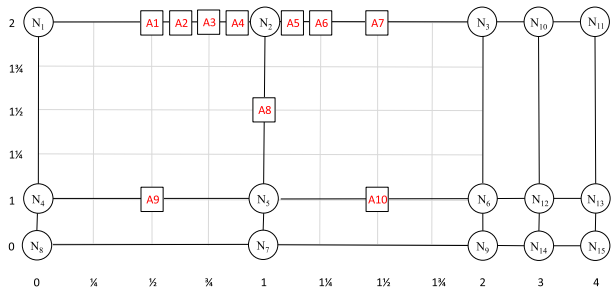
Algorithm 8 presents the pseudocode for the Significant Route Miner with basic Dynamic Segmentation (SRM_D). All the edges with a activities (where $a > 0$) are split into a nodes and $a + 1$ edges (line 2). The weights of the new edges in the segmented network are then updated based on the distance 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 segmentation assumes that the locations for all activities are known). Each new node that is formed (which was really an activity) is noted and stored in $dynamicNodes$ (line 5). The shortest paths between each pair of nodes in $dynamicNodes$ are then calculated using likelihood pruning. Steps 2 - 4 of SRM_D are the same as SRM_GIS.

Algorithm 8 Significant Route Miner with Dynamic Segmentation (SRM_D)

```
Inputs and Outputs are the same as NaïveSRM
Algorithm:
   {Step 1:  Calculate shortest paths between activities}
1: for each edges e ∈ 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 ← n_a
5: P ← shortest paths between all pairs of nodes ∈ dynamicNodes based on likelihood pruning
6: Steps 2 - 4 same as SRM_GIS
```

Dynamic Segmentation Example: Figure 4.7 illustrates a spatial network that has been dynamically segmented. In this case the number of nodes exceeds the number of activities in the spatial network. The algorithm proceeds by creating new nodes and edges based on the locations of the activities. For example, activities A1, A2, A3, etc. become nodes in the spatial network. As a result, the algorithm is able to evaluate paths that start and end at each activity. The figure illustrates a few sample paths in the dynamically segmented network, some of which would not have otherwise been evaluated. An example of such a path is $\langle A1, A2, A3, A4, A5, A6, A7\rangle$ with a likelihood ratio of 49. In a statically segmented network, only paths comprised of the original nodes and edges such as $\langle N1, N2, N3\rangle$ (with a likelihood ratio of 23.33) would be evaluated. Dynamic segmentation allows us to evaluate shortest paths we would not have otherwise considered. Because the edge weights are based on the locations of the activities, the likelihood ratios tend to be more precise (since the extra portions of

the path before the first activity and after the last activity are trimmed). For example, $\langle A1, A2, A3, A4, A5, A6, A7\rangle$ and $\langle N1, N2, N3\rangle$ have the same set of activities but the weight of $\langle A1, A2, A3, A4, A5, A6, A7\rangle$ is less. In this case, the likelihood ratio for $\langle A1, A2, A3, A4, A5, A6, A7\rangle$ ends up being much higher.



Start	End	Path	a	w	λ
A1	A4	<A1,A2,A3,A4>	4	0.375	38.44
A5	A7	<A5,A6,A7>	3	0.375	24.71
A1	A7	<A1,A2,A3,A4,N2,A5,A6,A7>	7	1	49
N1	N3	<N1,N2,N3>	7	2	23.33

Figure 4.7: Example of Dynamic Segmentation where the network structure is altered such that new nodes are formed at the locations of activities and new edges are added to connect these nodes. Paths such as $\langle A1, A2, A3, A4, A5, A6, A7\rangle$ may be evaluated directly for statistical significance. In this example, the number of nodes exceeds the number of activities.

4.4.2 Algorithmic Refinements

Hierarchical Filter

The idea behind the hierarchical filter for dynamic segmentation is to mitigate the cost of evaluating shortest paths between activities when the number of activities exceed the number of nodes in the original spatial network (as this would introduce many new nodes and may result in many more shortest paths being evaluated). The pseudocode for the hierarchical filter may be found in Algorithm 9 (lines 2 - 10).

The algorithm first checks whether the number of activities exceeds the number of nodes (line 1) or vice-versa (line 11). If the number of activities exceeds the number of nodes in the spatial network, then dynamic segmentation may result in potentially many more shortest paths being evaluated (relative to all pair shortest paths on a statically

segmented network). In order to mitigate this cost, the algorithm seeks to avoid directly calculating all pair shortest paths between activities. Instead the algorithm attempts to determine these paths by stitching together shortest paths between nodes in the statically segmented network P_N with paths between activities and the start and end of paths in P_N (lines 2-10). The key in this case is storing the paths between each activity and the start and end nodes of its original edge e while e is being dynamically segmented (line 7).

On the other hand, if the number of nodes is greater than or equal to the number of activities in the spatial network, the dynamic segmentation process continues relatively straightforwardly. All the edges with a activities (where $a > 0$) are split into a nodes and $a + 1$ activities (line 13). The weights of the new edges in the segmented network are then updated based on the distance between activities (line 14). 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. Each new node that is formed (which was really an activity) is noted and stored in *dynamicNodes* (line 15). The shortest paths between each pair of nodes in *dynamicNodes* are then calculated using likelihood pruning.

Hierarchical Filter Example: Figure 4.8(a) illustrates an example of hierarchical filtering. Shortest path $\langle A1, A10\rangle$ is calculated by stitching together $\langle A1, N2\rangle$, $\langle N2, N5\rangle$, and $\langle N5, A10\rangle$ (ensuring that the resulting path is still a shortest path). During the stitching process, all combinations of possible paths are considered. For example, for A1, the shortest path to A10 could have gone through either N1 or N2, and for A10 the shortest path could have come through N5 or N6. All these combinations are considered.

Active Node Filter

Definition 15. ***Active node:*** *A node is considered active if at least one of its incident edges has one or more activities.*

In Figure 4.4, all nodes are active except N_7. In Figure 4.7, only nodes N_1 to N_6 are active.

A further refinement to the hierarchical filter is the active node filter, found on line 2 of Algorithm 9. The basic idea is that when activities exceed nodes, instead

of calculating all pair shortest paths between the original nodes, only shortest paths between active nodes in the network are calculated.

Active Node Filter Example: Figure 4.8(b) shows an example of the active node filter, which enhances the hierarchical filter by only considering paths between active nodes (as they connect edges with activities). Instead of considering all $\binom{7}{2} = 42$ paths between the original nodes shown in the figure, the active node filter will only consider the 30 paths between the original active nodes.

Algorithm 9 Significant Route Miner with Dynamic Segmentation using the Hierarchical Filter and Active Node Filter (SRM_TKDE)

```
Inputs and Outputs are the same as NaïveSRM
Algorithm:
   {Step 1:  Calculate shortest paths between activities}
1: if |A| > |N| then
2:    P_{N_A} ← shortest paths between active nodes in G
3:    for each edges e = (u, v) ∈ G with a > 0 activities do
4:       Split e into a nodes, n_a, and a + 1 edges, e_a
5:       update weights of e_a based on coordinates of activities
6:       dynamicNodes ← n_a
7:       update paths x, u and x, v for each activity x
8:    for each x_i ∈ A do
9:       for each x_j ∈ A do
10:          P ← shortestpath(x_i, x_j) based on combining shortest paths in P_N with the paths
   between each activity x_i and x_j and the nodes of their original edge
11: else
12:    for each edges e ∈ G with a > 0 activities do
13:       Split e into a nodes, n_a, and a + 1 edges, e_a
14:       update weights of e_a based on coordinates of activities
15:       dynamicNodes ← n_a
16:    P ← shortest paths between all pairs of nodes ∈ dynamicNodes based on likelihood pruning
17: Steps 2 - 4 same as SRM_GIS
```

Table 4.1 summarizes the number of paths explored by SRM with and without dynamic segmentation. An example of network 1 that is presented in the table may be seen in Figure 4.7 and an example of network 2 is illustrated in Figure 4.4. If the number of nodes exceeds the number of activities, fewer shortest paths will be explored when dynamic segmentation is invoked. If the number of activities exceeds the number of nodes, the hierarchical and active node filters facilitate the exploration of fewer shortest paths relative to a basic dynamic segmentation approach (where ϵ_1 and ϵ_2 are the costs of stitching together the paths).

Table 4.1: Number of paths explored by SRM with and without dynamic segmentation

	# Paths Explored	
	Network 1 (15 nodes,10 activities)	Network 2 (7 nodes, 10 activities)
SRM_GIS (static segmentation)	15 x 14 = 210 paths	7 x 6 = 42
SRM_D (proposed dynamic segmentation)	10 x 9 = 90 paths	10 x 9 = 90 paths
SRM_H (dynamic segmentation with the hierarchical filter)	10 x 9 = 90 paths	42 paths + ε_1
SRM_TKDE (dynamic segmentation with the active node filter)	10 x 9 = 90 paths	30 paths + ε_2

4.4.3 Theoretical Analysis

Theorem 7. *Dynamic Segmentation via Algorithm 8 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.*

Proof. 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 8 is correct. □

Theorem 8. *The hierarchical filter will not lose any significant shortest path p between activities if p is not a subset of any other significant shortest path between the activities.*

Proof. This may be argued in a similar manner to that of hierarchical routing [72]. 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 before dynamic segmentation. Therefore, we may only consider the shortest path between these nodes and append them to the shortest paths between each activity and the nodes of the original edge it was on, yielding the overall shortest path. Since all the shortest paths between activities and the start and end nodes of their original edge are considered and all the shortest paths between all nodes in the spatial network are considered, the hierarchical filter will not lose any significant shortest path p between activities if p is not a subset of any other significant shortest path between the activities. □

Lemma 8. *The active node filter will not lose any significant shortest path p between*

activities if p is not a subset of any other significant shortest path between the activities.

Proof. The set of optimal solutions in a dynamically segmented network all start and end with activities. The active node filter does not filter out any optimal solutions because (1) it considers all shortest paths between nodes with at least one incident edge that has one or more activities (i.e., active nodes) and (2) it does not filter any activities. □

Theorem 9. *The active node filter is a correct method for calculating shortest paths between activities*

Proof. The active node filter algorithmic refinement for dynamic segmentation is correct because it will not lose any significant paths (Lemma 8) and it does not filter out any optimal solutions (since it considers all shortest paths starting and ending with an edge that has 1 or more activities). □

Theorem 10. *SRM_TKDE is faster than SRM_GIS when the number of nodes exceeds the number of activities in the spatial network.*

Proof. The computational costs of the algorithms presented stem from 1) the cost of calculating all pair shortest paths and 2) the cost of assessing statistical significance for all shortest paths in the spatial network. Recall that for NaïveSRM, the total cost is $(N^2 logN \times C_{\lambda_p}) + (m \times N^2 logN \times C_{\lambda_p})$, where N is the set of nodes, $N^2 logN$ is the cost of calculating shortest paths in the spatial network, C_{λ_p} is the cost of calculating the likelihood ratio of path p, and m is the number of Monte Carlo simulations.

The basic difference between SRM_GIS and SRM_TKDE stems from the set of nodes being considered: the set of original network nodes N versus the set of dynamic nodes N_D. For SRM_GIS, the total cost is $((N \times r_{\hat{\lambda}})^2 logN \times C_{\lambda_p}) + (m \times (N^2 logN \times r_m) \times C_{\lambda_p})$, where N is the set of nodes, $(N \times r_{\hat{\lambda}})^2 logN$ is the cost of calculating shortest paths for a set of shortest paths that is a superset of all paths in G with $\lambda_p \geq \theta$, $r_{\hat{\lambda}}$ (whose value is between 0 and 1) is the ratio of shortest paths with $\lambda_p \geq \theta$ to all shortest paths, C_{λ_p} is the cost of calculating the likelihood ratio of path p, m is the number of Monte Carlo simulations, and r_m (whose value is between 0 and 1) is the ratio of shortest paths calculated before finding a path whose likelihood beats the maximum likelihood in the original graph to all shortest paths.

The total cost of SRM_TKDE is $((N_D \times r_{\hat{\lambda}})^2 logN_D \times C_{\lambda_p}) + (m \times (N_D^2 logN_D \times r_m) \times C_{\lambda_p})$, where N_D is the set of dynamic nodes (i.e., activities), $(N \times r_{\hat{\lambda}})^2 logN$ is the cost of calculating shortest paths for a set of shortest paths that is a superset of all paths in G with $\lambda_p \geq \theta$, $r_{\hat{\lambda}}$ (whose value is between 0 and 1) is the ratio of shortest paths with $\lambda_p \geq \theta$ to all shortest paths, C_{λ_p} is the cost of calculating the likelihood ratio of path p, m is the number of Monte Carlo simulations, and r_m (whose value is between 0 and 1) is the ratio of shortest paths calculated before finding a path whose likelihood beats the maximum likelihood in the original graph to all shortest paths. The hierarchical and active node filters are applied when the number of activities exceeds the number of nodes in the spatial network. In this case, the cost becomes $(((N_A \times r_{\hat{\lambda}})^2 logN_A + C_{NA}) \times C_{\lambda_p}) + (m \times (N_A^2 logN_A \times r_m) \times C_{\lambda_p})$, where N_A is the set of active nodes and C_{NA} is the cost of appending shortest paths between the original nodes in the network to the shortest paths between each activity and the nodes joining the original edge that it was on. □

4.5 Case study

We conducted a qualitative evaluation of our algorithm with and without dynamic segmentation (SRM_TKDE vs. SRM_GIS), comparing their analysis with the analysis of SaTScan [69] (continuous Poisson process) on a real pedestrian fatality data set [1], shown in Figure 4.9(a). As noted earlier, SaTScan discovers areas of significant activity that are represented as circles on the spatial network while SRM_GIS 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 overall activity, and weight was the actual road network distance. The maps were prepared using QGIS' Open Layers plugin [73], and the road network was from the US Census Bureaus TIGER/Line Shapefiles [50].

When evaluating the techniques, we considered the outputs of circles vs. shortest paths. While p-value thresholds of 0.05 or lower are often desired, we used a p-value threshold of 0.15 because the circles chosen by SaTScan had high p-values for this dataset. As noted earlier, pedestrian fatalities usually occur on streets, particularly

along arterial roadways [2]. Thus this activity can be said to have a linear generator. However, the results generated by SaTScan do not capture this. From Figure 4.9(b), it is clear that the circle-based output is meant for areas, not streets. In contrast, the shortest paths detected by SRM_GIS fully capture the significant activities on the arterial roads (some of the paths in Figure 4.9(c) are overlapping). Furthermore, the paths in the figure make sense in this context due to the inherently linear nature of the activities.

We also compared the output of our miner with dynamic segmentation (SRM_TKDE) to the outputs of SaTScan and SRM_GIS (our miner without dynamic segmentation). As we can observe from Figure 4.9(d), relative to SaTScan, SRM_TKDE is able to capture significant activities on the arterial roads (just as SRM_GIS). In contrast to SRM_GIS, SRM_TKDE is also able to detect paths occurring on the sub-edge level such as the blue path (top-center of the figure). Therefore, even if there were paths that were not significant because the activities on them were on long edges, if the activities were close to each other on the network, they may show up in SRM_TKDE's result.

4.6 Experimental Evaluation

The goal of our experiments was to evaluate the scalability of the proposed approach by varying and observing the effect of four workload parameters: number of nodes, likelihood ratio threshold θ, p-value threshold, and number of activities. All experiments were performed on a Mac Pro with a 2 x Xeon Quad Core 2.26 GHz processor and 16 GB RAM. For each workload experiment we compared the following candidates:

- Naïve Significant Route Miner (NaïveSRM)
- Significant Route Miner with likelihood pruning and Monte Carlo speedup (SRM_GIS) [68]
- Significant Route Miner with Dynamic Segmentation using the Hierarchical and Active Node Filters (SRM_TKDE)

4.6.1 Experiment Data Sets

Our experiments varying nodes, likelihood ratio threshold θ, and p-value threshold were performed on real-world data obtained from the Fatality Analysis Reporting System (FARS) encyclopedia [1]. The dataset contained geospatial and temporal data describing 487 pedestrian fatalities in Orange County, FL (which includes Orlando), from 2001 to 2011. For each edge (portion of road) in the network, fatality count was aggregated, yielding overall activity, and weight was the actual road network distance. The road network was obtained from the US Census Bureau's TIGER/Line Shapefiles [50]. Our experiment varying number of activities was performed on a synthetic dataset. The synthetic dataset was generated under the null hypothesis. The road network used in the synthetic dataset was the same as that used in the real-world dataset.

4.6.2 Experimental Results

Effect of the Number of Nodes: We used subsets of the real dataset. We varied the number of nodes from 500 to 2500, which is akin to varying the number of shortest paths (routes) from $250,000$ to $6,250,000$ (since there are $\binom{n}{2}$ shortest paths in the spatial network). We set the p-value threshold to 0.05, the number of Monte Carlo simulations to 100, and the likelihood ratio threshold θ to 20. The number of activities ranged from 40 for 500 nodes to 159 for 2500 nodes. Figure 4.10(a) gives the execution times. As can be seen, SRM_TKDE is fastest, followed by SRM_GIS, as the number of nodes vastly exceeded the number of activities. For SRM_TKDE, only shortest paths between activities are evaluated because of dynamic segmentation. In this case, since the number of activities is much lower than the number of nodes, we observe a large computational speedup. Computational savings increases as the number of nodes increases due to Likelihood Pruning, Monte Carlo Speedup, and Dynamic Segmentation.

Effect of the Likelihood Ratio Threshold θ: We used subsets of the real dataset. The p-value was set to 0.05, the number of Monte Carlo simulations was set to 100, and the number of nodes was set to 1000. The number of activities was 68. Figure 4.10(b) gives the execution times. Again, SRM_TKDE is fastest, followed by SRM_GIS, as the number of nodes vastly exceeded the number of activities. Computational savings increases as the likelihood ratio increases due to Likelihood Pruning, Monte Carlo

Speedup, and Dynamic Segmentation.

Effect of the P-value: We used subsets of the real dataset. The number of nodes was set to 500, the likelihood ratio threshold θ was set to 20, and the number of Monte Carlo simulations was set to 100. The number of activities was 40 for 500 nodes. Figure 4.10(c) gives the execution times. As can be seen, SRM_TKDE is fastest, followed by SRM_GIS, as the number of nodes vastly exceeded the number of activities. Computational savings increases as the p-value increases due to Likelihood Pruning, Monte Carlo Speedup, and Dynamic Segmentation.

Effect of the Number of Activities: This experiment was carried out on the synthetic dataset. The number of nodes was set to 500, the likelihood ratio threshold θ was set to 2, and the number of Monte Carlo simulations was set to 100. We varied the number of activities from 200 to 800. When the number of activities is 800, dynamic segmentation increases the network size by adding 800 new nodes to the 500 nodes in the original network. Figure 4.10(d) gives the execution times. When the number of nodes greatly exceeded the number of activities, SRM_TKDE was fastest, as predicted by the cost models. A crossover occurred when the number of activities exceeded the number of nodes (at 800 activities). As can be observed, SRM_TKDE still performed on par with NaïveSRM and SRM_GIS while considering routes at the sub-edge level, a feature not present in NaïveSRM or SRM_GIS [68].

4.7 Discussion and Future Work

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 [6], K-Means [10], KMR [11], and Maximum Subgraph Finding [70]. For example, the algorithm from our previous work [11] on summarizing activities using routes may return routes that are not statistically significant. Figure 4.11 shows an example where DBScan [6] returns 7 chance clusters 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.

Alternative network footprints: Summarizing significant network footprints of activities may be done using significant subgraphs, significant paths, significant shortest paths, minimum spanning trees, etc. Each representation entails a tradeoff between fidelity and computational scalability. For example, subgraphs may offer accurate significant network footprints but their calculation may be computationally intensive due to their exponential number. As an initial step, we have selected shortest paths to summarize significant network footprints of activities. While shortest paths may lose some fidelity, they offer computational scalability because their number is bounded (i.e., $\binom{n}{2}$, where n is the number of nodes). The union of shortest paths may also be used to represent other network footprints.

Multi-Scale Model: Different regions of the network may require different algorithmic refinements. For example, linear hotspots may be short on residential streets, medium length on county roads, and long on interstate highways. Modeling these differences is important for improving solution quality. A potential approach towards this end is a multi-scale model, since roads in transportation networks are often categorized into highways, county roads, residential streets. etc. We plan to investigate a multi-scale approach for significant linear hotspot discovery in future work.

P-value and the Study Area Size: It is known that the likelihood ratio and p-value are sensitive to the size of the study area. In other words, as we enlarge or decrease the network size (study area) to either include or remove empty paths or edges, the likelihood ratio and p-value will be impacted accordingly. We plan to investigate this sensitivity in future work and explore methods for mitigating or addressing it.

Alternate Problem Formulations: In an alternate formulation of the problem, the spatial network may be modeled with an activity count function $a(u) \geq 0$ for each node. The idea is that activities may also occur at nodes in addition to being distributed within network edges. In this way, the current approach may be extended to capture activities at nodes (e.g., vehicle accidents). If activities are modeled as counts at each node, this may alter the computational structure. We plan to investigate this in future work.

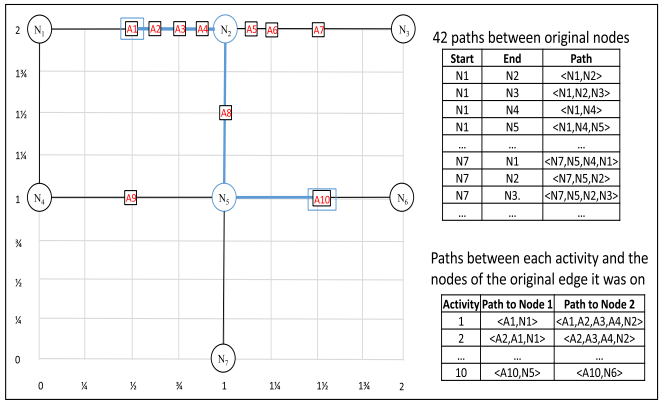
In another formulation of the problem, one may require every edge with significant linear hotspots to meet certain criteria (e.g., minimum likelihood ratios and p-values).

This alternate problem formulation may exhibit a different computational structure (e.g., monotonicity) and allow more aggressive bottom-up pruning that is popular in the graph mining literature. Here the results may be significantly different from those in the current problem formulation.

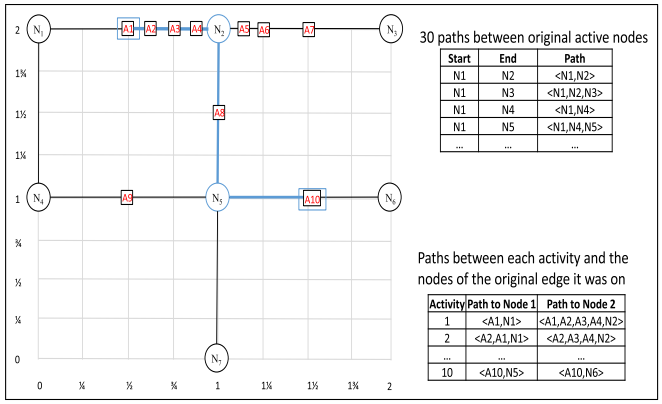
In future work, we plan to explore other types of data that may not be associated with a point in a street (e.g., aggregated pedestrian fatality data at the zip code level). The present research is centered on finding high concentrations of activities whose counts and locations are deterministic. However, 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. In the current approach we permuted the activities. Alternatively we can permute activity count which may provide a Poisson distribution assumption. We plan to investigate this in future work. Finally, incorporating time [74, 75, 76] and investigating the underlying factors of significant linear hotspots will be explored.

4.8 Conclusion

This work explored the problem of significant linear hotspot discovery in relation to important application domains such as preventing pedestrian fatalities and crime analysis. We proposed a significant route miner that discovers multiple statistically significant shortest paths at the sub-edge level in a spatial network. The proposed approach uses Likelihood Pruning, Monte Carlo Speedup, and Dynamic Segmentation to enhance its performance and scalability. We presented a case study comparing our proposed method with SaTScan on pedestrian fatality data. Experimental evaluation using real-world and synthetic data indicated that the algorithmic refinements utilized by our approach yielded substantial computational savings without sacrificing result quality.

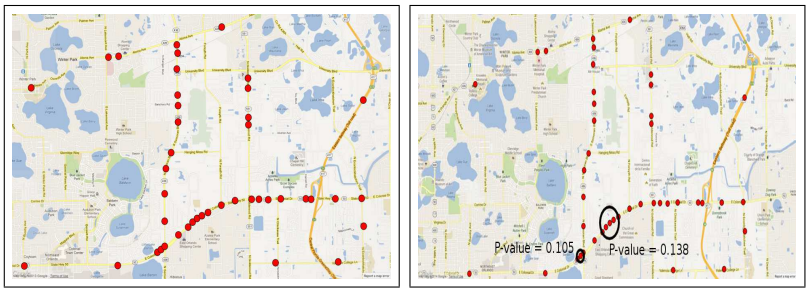


(a)



(b)

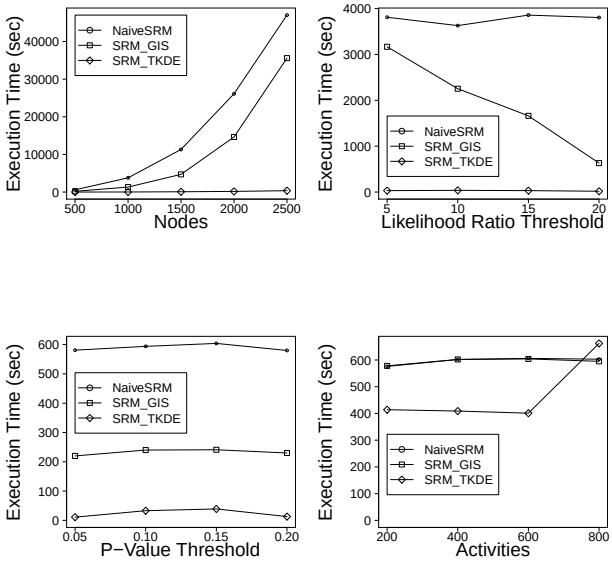
Figure 4.8: Example of (a) the hierarchical filter and (b) the active node filter for dynamic segmentation. Shortest paths between activities are determined by stitching together (1) shortest paths between nodes in the statically segmented network with (2) paths between activities and the start and end of their original edges. Shortest path $\langle A1, A10 \rangle$ is calculated by stitching together $\langle A1, N2 \rangle$, $\langle N2, N5 \rangle$, and $\langle N5, A10 \rangle$. The active node filter refines (1) by only considering shortest paths between active nodes. (Best in color).



(a) Input (b) SaTScan

(c) SRM_GIS (d) SRM_TKDE

Figure 4.9: Comparing SRM_GIS (without dynamic segmentation), SRM_TKDE (with dynamic segmentation), and SaTScan's output for a p-value threshold of 0.15 and $\theta = 1.75$ on pedestrian fatality data from Orlando, FL [1] (Best in color).



(c) P-value threshold (d) Activities

Figure 4.10: Scalability with increasing (a) number of nodes, (b) likelihood ratio threshold θ, (c) p-value threshold, and (d) activities

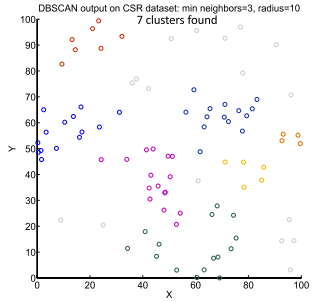


Figure 4.11: Colored dots are part of chance clusters identified by DBScan [6] on a complete spatial randomness dataset (Best in color).

Chapter 5

Conclusions and Future Work

Summarizing spatial network-based observations is important in several application domains such as transportation safety, public safety, public health, and disaster response. For example, transportation planners and engineers may need to identify road segments that pose risks for pedestrians and require redesign, epidemiologists may try to understand the spread of disease so that patterns of progression can be established, and hydrologists may try to summarize significant environmental change on water resources to understand the behavior of river networks and lakes. However, summarizing spatial network-based observations is computationally challenging for the following reasons: (1) There may be a large number of k-subsets of connected components in the network, (2) Patterns may not obey the monotonicity property, and (3) There may be a large number of candidates.

In this thesis, we proposed a suite of techniques that addressed these three computational challenges. First, spatial network activity summarization was explored that addressed the challenge of a large number of k-subsets of connected network components by proposing the K-Main Routes (KMR) approach. Second, geo-referenced time-series summarization was studied to address the challenge where patterns may not obey the monotonicity property by proposing the K-Full Tree (KFT) technique. Finally, the challenge of the large number of candidates was examined by solving the significant route discovery problem using our proposed Significant Route Miner approach.

A summary of the key results in this research are presented in Section 5.1 and future directions of this thesis are presented in Section 5.2.

5.1 Key Results

This section presents a summary of the major results that were produced as a part of this thesis.

- **Spatial Network Activity Summarization:** In spatial network activity summarization (SNAS), we are given a spatial network and a collection of activities (e.g., pedestrian fatality reports, crime reports) and the goal is to find k shortest paths that summarize the activities. SNAS is important for applications where observations occur along linear paths such as roadways, train tracks, etc. SNAS is computationally challenging because of the large number of k subsets of shortest paths in a spatial network. Previous work has focused on either geometry or subgraph-based approaches (e.g., only one path), and cannot summarize activities using multiple paths. This work proposes a K-Main Routes (KMR) approach that discovers k shortest paths to summarize activities. KMR generalizes K-means for network space but uses shortest paths instead of ellipses to summarize activities. To improve performance, KMR uses network Voronoi, divide and conquer, and pruning strategies. We present a case study comparing KMR's network-based output (i.e., shortest paths) to geometry-based outputs (e.g., ellipses) on pedestrian fatality data. Experimental results on synthetic and real data show that KMR with our performance-tuning decisions yields substantial computational savings without reducing summary path coverage.

- **Geo-referenced Time-Series Summarization:** Given a set of regions with activity counts at each time instant (e.g., a listing of countries with number of mass protests or disease cases over time) and a spatial neighbor relation, geo-referenced time-series summarization (GTS) finds k-full trees that maximize activity coverage. GTS has important potential societal applications such as understanding the spread of political unrest, disease, crimes, fires, pollutants, etc. However, GTS is computationally challenging because (1) there are a large number of subsets of k-full trees due to the potential overlap of trees and (2) a region with no activity may be a part of a larger region with maximum activity coverage, making apriori-based pruning inapplicable. Previous approaches for spatio-temporal data mining detect anomalous or unusual areas and do not summarize activities. We propose a

k-full tree (kFT) approach for GTS which features an algorithmic refinements that leads to computational savings without affecting result quality. The algorithmic refinement is voronoi partition assignment for partitioning regions. Analytical and experimental results show that the algorithmic refinement substantially reduces computational cost. We also present a case study that shows the output of our approach on Arab Spring data.

- **Significant Linear Hotspot Discovery:** Given a spatial network and a collection of activities (e.g., pedestrian fatality reports, crime reports), Significant Linear Hotspot Discovery (SLHD) finds shortest paths in the spatial network where the concentration of activities is unusually high (i.e., statistically significant). SLHD is important for societal applications in transportation safety, public safety, or public health such as finding paths with significant concentrations of accidents, crimes, or diseases. SLHD is challenging because 1) there are a potentially large number of candidate paths ($\sim 10^{16}$) in a given dataset with hundreds of millions of activities or network nodes and 2) significance testing does not obey the monotonicity property. SaTScan may miss many significant paths since a large fraction of the area bounded by circles for activities on a path will be empty. Previous network-based approaches only consider a small fraction of the network and only one significant network component (e.g., path). We propose novel algorithms for discovering statistically significant linear hotspots using the ideas of likelihood pruning, Monte Carlo speedup, and dynamic segmentation. We present a case study comparing the proposed approach with existing techniques on real data. Experimental results show that the proposed algorithm, with our algorithmic refinements, yields substantial computational savings without reducing result quality.

5.2 Future Directions

We organize the future directions of this thesis into two categories: (a) short-term directions and (b) long-term directions.

Table 5.1: Summarization framework with future work (Best in Color)

Data Genre (Domain)	Group Definition	Group Representation	Interest Measure
Relational Table (a set of rows)	- A partition of rows - Significant groups of rows	- Attribute values (e.g., age-group)	- aggregate property - max coverage (e.g., count) - min sum of distances Ch. 5.2.1 - significance metric (e.g., p-value, likelihood ratio, confidence interval) - Computational metric - sequential response time - speedup on parallel platform (e.g., MapReduce) Ch. 5.2.1
Vector Space	- A partition of vectors - Significant groups of vectors	- A vector, basis function	
Spatial (Euclidean Space)	- A partition of space - Significant groups of sub-space	- points, polygons, ellipses, line-strings	
Spatial Network	- A partition of a network Ch. 2 - Significant groups of sub-networks Ch. 4	- path - domain concepts (e.g.,natural geographic features) Ch. 5.2.2	
Geo-referenced Time-series (GT)	- A partition of a GT Ch. 3 - Significant groups of sub-GTs Ch. 5.2.2	- ST-full tree	

before thesis thesis future work

5.2.1 Short-term Directions

We plan to explore the following two short term directions a) Parallelizing Spatial Network Summarization and b) Activity-based Network Summarization for Disaster Response.

Parallelizing Spatial Network Summarization

As the size of spatial networks increases and the number of observations become more numerous, parallelization becomes necessary in achieving computational scalability. However, parallelizing K-Main Routes (KMR), K-Full Tree (KFT), the Smart Significant Route Miner (SmartSRM), and other iterative techniques is challenging because these algorithms use previous information for the next iteration. Although processing one iteration is parallelizable, the synchronization overhead across iterations for cloud environments is too enormous to maintain speedups. We plan to evaluate approaches such as Spark [77], which has a cheaper "Reduce" step, with iterative GIS workloads. Our future work will also include non-iterative algorithms or different parallel programming models. Finally, recent small diameter social graph processing tools, e.g., Pregel [78], will be evaluated for larger diameter spatial networks, e.g., roadmaps.

Activity-based Network Summarization for Disaster Response

Novel computational techniques are needed for enhancing situational awareness for disaster response, which is a national imperative [79]. Beyond the unquantifiable costs of injury and loss of life from disasters, economic damages from Hurricane Sandy alone in the United States exceeded $60 billion [80]. Disaster response during emergency management includes action taken immediately after a disastrous event with the aim of saving life, protecting property, and dealing with immediate disruption, damage, or other effects caused by the disaster (e.g., hurricanes, earthquakes, terrorist attacks, etc.) [81, 82]. The 2010 Haiti earthquake saw numerous requests for relief including requests for food, water, and medicine. Emergency managers are tasked with making resource allocation decisions based on the locations of the affected population.

Exploring spatial network activity summarization and proposing the K-Main Routes algorithm [18] was an initial step towards assisting emergency managers. However, several outstanding issues remain while attempting to further improve situational awareness for disaster response, which will shape future research. For example, we plan to investigate a distance-based rather than coverage-based objective function where the distance between activities and summary routes is minimized to reduce how far the affected population has to walk. We also plan to extend our approach to account for different types of activities. A building collapsing on someone might require more immediate attention than other types of activities and should be summarized accordingly.

5.2.2 Long-term Directions

In the long term, the results of this thesis pave the way for designing new summarization techniques that explore a) Statistical Significance and b) Domain Concepts and Theory-aware Data Summarization.

Statistical Significance

In the summarization framework presented in Table 1.1, statistical significance emerges as a major theme across summarizing all genres of data. Statistical significance is important for finding summaries which are not due to just chance alone. Open areas of research include finding statistically significant ellipses, trees, and subgraphs as potental

group representatives. For example, we consider the problem of geo-referenced time-series summarization using significant trees as one possible long term direction under the broader umbrella of exploring statistical significance.

Given a set of regions with activity counts at each time instant (e.g., a listing of countries with number of mass protests or disease cases over time) and a spatial neighbor relation, geo-referenced time-series summarization using significant trees (GTSST) finds trees whose activity count is unusually high (i.e., statistically significant). GTSST has important potential societal applications such as understanding the spread of political unrest, disease, crimes, fires, pollutants, etc. However, GTSST is computationally challenging because (1) there are a large number of subsets of trees due to the potential overlap of trees and (2) significance testing does not obey the monotonicity property. Previous approaches for spatio-temporal data mining do not summarize regions with a significant number of activities. We propose a Spatio-Temporal Significant Tree (STST) approach for GTSST. We plan to validate STST using analytical evaluation, case studies, and experimental evaluation.

Domain Concepts and Theory-aware Data Summarization

Concepts or theories from particular domains may play a critical role in data summarization and as such we advocate a complementary approach: Domain Concepts and Theory-aware Data Summarization. This approach aims to create a symbiotic interplay between concepts and theories from specific domains (e.g., environmental criminology, epidemiology, climate science) and data summarization.

For example, the domain of environmental criminology [16] proposes several theories such as Routine Activity Theory (RAT) and Rational Choice Theory (RCT). RAT suggests that the location of a crime is related to the criminals frequently visited areas. In RCT, man is considered a reasoning actor who weighs costs and benefits and makes a rational choice. As such there exists a constant tension between the offenders desire to divert attention from his or her home base and the desire to travel no further than necessary to commit crimes [7]. Leveraging such domain information may facilitate the discovery of new types of summaries such as ring-shaped summaries, as illustrated in Figure 5.1.

In epidemiology, traditional summarization techniques such as hotspot detection or

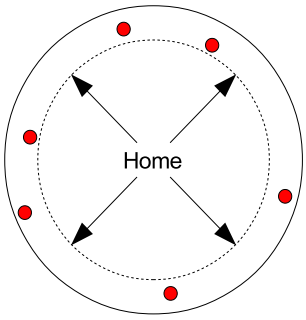


Figure 5.1: Ring-Shaped Summary based on domain concepts and theories in Environmental Criminology. There exists a constant tension between the offenders desire to divert attention from his or her home base and the desire to travel no further than necessary to commit crimes [7] (Best in Color)

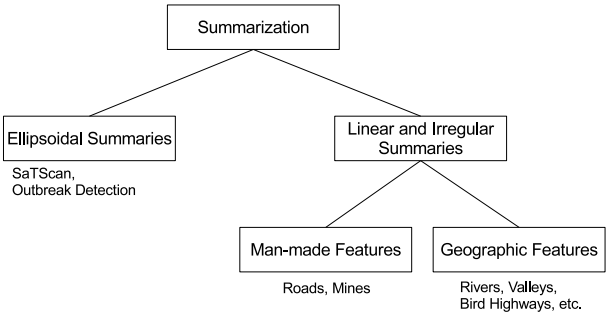


Figure 5.2: Classification of Summarization featuring Natural Geographic Concepts

disease outbreak detection from point data all focus on discovering traditional summaries, e.g., circle or ellipsoidal patterns. However, as outlined in Figure 5.2, summaries may also be based on geographic features such as detecting outbreaks along rivers, valleys, bird highways, etc. We propose to investigate summaries based on natural geographic concepts (e.g., finding outbreaks along geographic features as compared to administrative boundaries such as counties). We also plan to assess the tradeoff between natural geographic summaries and traditional summaries in terms of fidelity and computational scalability.

References

[1] Fatality Analysis Reporting System (FARS) Encyclopedia, National Highway Traffic Safety Administration (NHTSA), http://www.nhtsa.gov/FARS.

[2] 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.

[3] Arab Spring. `en.wikipedia.org/wiki/Arab_Spring/`.

[4] 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.

[5] 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.

[6] 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.

[7] Ronald Victor Gemuseus Clarke. *Situational crime prevention*. Criminal Justice Press, 1997.

[8] Ramez Elmasri. *Fundamentals of database systems*. Pearson Education India, 2008.

[9] 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.

[10] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[11] 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.

[12] P.J. Brantingham and P.L. Brantingham. *Environmental criminology*. Sage Publications Beverly Hills, CA, 1981.

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

[14] M.S. Scott and United States. Dept. of Justice. Office of Community Oriented Policing Services. *Assaults in and around bars*. US Department of Justice, Office of Community Oriented Policing Services, 2001.

[15] L.E. Cohen and M. Felson. Social change and crime rate trends: A routine activity approach. *American sociological review*, pages 588–608, 1979.

[16] PJ Brantingham and PL Brantingham. Environment, routine and situation: Toward a pattern theory of crime. *Advances in criminological theory*, 5:259–294, 1993.

[17] JE Eck, S. Chainey, JG Cameron, M. Leitner, and RE Wilson. Mapping crime: Understanding hot spots. 2005.

[18] Dev Oliver, Shashi Shekhar, J Kang, Renee Laubscher, Veronica Carlan, and Abdussalam Bannur. A k-main routes approach to spatial network activity summarization. *to appear in IEEE Transactions on Knowledge and Data Engineering*.

[19] Dev Oliver, Shashi Shekhar, James M Kang, Renee Laubscher, Veronica Carlan, and Michael R Evans. Geo-referenced time-series summarization using k-full trees: A summary of results. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pages 797–804. IEEE, 2012.

[20] Dev Oliver, Shashi Shekhar, Xun Zhou, Emre Eftelioglu, Michael R. Evans, Qiaodi Zhuang, J Kang, Renee Laubscher, and Christopher Farah. Significant route discovery: A summary of results. *to appear in the Eighth International Conference on Geographic Information Science*.

[21] 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.

[22] Huffington Post, Hungary: Snowstorm strands thousands in their cars (Accessed: April 16, 2013).

[23] Chicago Tribune, Metra argues for delay of 'fail-safe' rail system, http://www.chicagotribune.com/news/local/ct-met-metra-collision-prevention-20130409,0,3710438.story.

[24] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley Online Library, 1990.

[25] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–144. Citeseer, 1994.

[26] M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88, 2004.

[27] R.G. D'Andrade. U-statistic hierarchical clustering. *Psychometrika*, 43(1):59–67, 1978.

[28] M. Celik, S. Shekhar, B. George, J.P. Rogers, and J.A. Shine. Discovering and quantifying mean streets: A summary of results. Technical report, Technical Report 07-025, University of Minnesota, Computer Science and Engineering, 2007.

[29] K. Buchin, S. Cabello, J. Gudmundsson, M. Löffler, J. Luo, G. Rote, R. I. Silveira, B. Speckmann, and T. Wolle. Detecting Hotspots in Geographic Networks. *Advances in GIScience*, pages 217–231.

[30] S.A. Roach. *The theory of random clumping*. Methuen, 1968.

[31] A. Okabe, K.I. Okunuki, and S. Shiode. The SANET toolbox: New methods for network spatial analysis. *Transactions in GIS*, 10(4):535–550, 2006.

[32] S. Shiode and A. Okabe. Network variable clumping method for analyzing point patterns on a network. In *Unpublished paper presented at the Annual Meeting of the Associations of American Geographers, Philadelphia, Pennsylvania*, 2004.

[33] K. Aerts, C. Lathuy, T. Steenberghen, and I. Thomas. Spatial clustering of traffic accidents using distances along the network. In *Proc. 19th Workshop of the International Cooperation on Theories and Concepts in Traffic Safety*, 2006.

[34] P.G. Spooner, I.D. Lunt, A. Okabe, and S. Shiode. Spatial analysis of roadside Acacia populations on a road network using the network K-function. *Landscape ecology*, 19(5):491–499, 2004.

[35] T. Steenberghen, T. Dufays, I. Thomas, and B. Flahaut. Intra-urban location and clustering of road accidents using GIS: a Belgian example. *International Journal of Geographical Information Science*, 18(2):169–181, 2004.

[36] I. Yamada and J.C. Thill. Local indicators of network-constrained clusters in spatial point patterns. *Geographical Analysis*, 39(3):268–292, 2007.

[37] S. Shiode and N. Shiode. Detection of multi-scale clusters in network space. *International Journal of Geographical Information Science*, 23(1):75–92, 2009.

[38] 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.

[39] M. Terrovitis, S. Bakiras, D. Papadias, and K. Mouratidis. Constrained shortest path computation. *Advances in Spatial and Temporal Databases*, pages 923–923, 2005.

[40] 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.

[41] T.H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[42] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman San Francisco, 1979.

[43] D.S. Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143. PWS Publishing Co., 1996.

[44] N. Levine. CrimeStat: A spatial statistics program for the analysis of crime incident locations (v 2.0). *Ned Levine & Associates, Houston, TX, and the National Institute of Justice, Washington, DC*, 2002.

[45] S. Borah and M.K. Ghose. Performance analysis of AIM-K-means & K-means in quality cluster generation. *Arxiv preprint arXiv:0912.3983*, 2009.

[46] A.R. Barakbah and Y. Kiyoki. A pillar algorithm for K-Means optimization by distance maximization for initial centroid designation. *IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Nashville-Tennessee*, 2009.

[47] S.S. Khan and A. Ahmad. Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters*, 25(11):1293–1302, 2004.

[48] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734. San Francisco, 2000.

[49] P.S. Bradley and U.M. Fayyad. Refining initial points for k-means clustering. In *Proc. 15th International Conf. on Machine Learning*, volume 727. Citeseer, 1998.

[50] US Census Bureau 2010 Census TIGER/Line Shapefiles. http://www.census.gov/geo/www/tiger/tgrshp2010/tgrshp2010.html.

[51] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo. Spatio-temporal clustering. *Data mining and knowledge discovery handbook*, pages 855–874, 2010.

[52] CIA World Factbook. `https://www.cia.gov/library/publications/the-world-factbook/`.

[53] Foreign Affairs. *The New Arab Revolt: What Happened, What It Means, and What Comes Next*. Council on Foreign Relations, 2011.

[54] Disease Surveillance. `http://en.wikipedia.org/wiki/Disease_surveillance/`.

[55] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[56] Simpson's paradox. `http://en.wikipedia.org/wiki/Simpson's_paradox`.

[57] M. Kulldorff. A spatial scan statistic. *Communications in statistics-theory and methods*, 26(6):1481–1496, 1997.

[58] M. Kulldorff, WF Athas, EJ Feurer, BA Miller, and CR Key. Evaluating cluster alarms: a space-time scan statistic and brain cancer in los alamos, new mexico. *American Journal of Public Health*, 88(9):1377–1380, 1998.

[59] M. Kulldorff, R. Heffernan, J. Hartman, R. Assunção, and F. Mostashari. A space–time permutation scan statistic for disease outbreak detection. *PLoS medicine*, 2(3):e59, 2005.

[60] D.B. Neill, A.W. Moore, M. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 218–227. ACM, 2005.

[61] M.A. Costa, M. Kulldorff, and R.M. Assuncao. A space time permutation scan statistic with irregular shape for disease outbreak detection. *Advances in Disease Surveillance*, 4:243, 2007.

[62] D. Joshi. *Polygonal Spatial Clustering*. PhD thesis, University of Nebraska, 2011.

[63] M. Worboys and M. Duckham. *GIS: A computing perspective*. CRC, 2004.

[64] 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.

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

[66] Luiz Duczmal and Renato Assuncao. A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Computational Statistics & Data Analysis*, 45(2):269–286, 2004.

[67] 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.

[68] 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 *To appear in the Eighth International Conference on Geographic Information Science Vienna - September, 23-26*, 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] 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.

[71] Shuchi Chawla and Tim Roughgarden. Single-source stochastic routing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 82–94. Springer, 2006.

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

[73] Quantum GIS OpenLayers Plugin (Accessed: January 23, 2014).

[74] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery: A summary of results. In *SDM*, pages 327–338. SIAM, 2010.

[75] 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.

[76] 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.

[77] `http://spark.incubator.apache.org/`. Apache Spark.

[78] G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 international conference on Management of data*, pages 135–146. ACM, 2010.

[79] Susan L Cutter, Joseph A Ahearn, Bernard Amadei, Patrick Crawford, Elizabeth A Eide, Gerald E Galloway, Michael F Goodchild, Howard C Kunreuther, Meredith Li-Vollmer, Monica Schoch-Spana, et al. Disaster resilience: A national imperative. *Environment: Science and Policy for Sustainable Development*, 55(2):25–29, 2013.

[80] Huffington Post, Sandy Economic Damage Worse Than Expected At $62 Billion (Accessed: February 28, 2014).

[81] W.N. Carter. *Disaster management: A disaster manager's handbook*. Asian Development Bank, 1991.

[82] Dev Oliver, Rupa Tiwari, Michael R. Evans, and Shashi Shekhar. Disaster Response and Relief, VGI Volunteer Motivation in. *to appear in the Encyclopedia of Social Network Analysis and Mining*.