Significant Linear Hotspot Discovery

Xun Tang, Emre Eftelioglu, Dev Oliver, and Shashi Shekhar

Abstract—Given a spatial network and a collection of activities (e.g., pedestrian fatality reports, crime reports), Significant Linear Hotspot Discovery (SLHD) finds all shortest paths in the spatial network where the concentration of activities is statistically significantly high. SLHD is important for societal applications in transportation safety or public safety such as finding paths with significant concentrations of accidents or crimes. SLHD is challenging because 1) there are a potentially large number of candidate paths ($\sim 10^{16}$) in a given dataset with millions of activities and road network nodes and 2) test statistic (e.g., density ratio) is not monotonic. Hotspot detection approaches on euclidean space (e.g., SaTScan) may miss significant paths since a large fraction of an area bounded by shapes in euclidean space for activities on a path will be empty. Previous network-based approaches consider only paths between road intersections but not activities. This paper proposes novel models and algorithms for discovering statistically significant linear hotspots using the algorithms of neighbor node filter, shortest path tree pruning, and Monte Carlo speedup. We present case studies comparing the proposed approaches with existing techniques on real data. Experimental results show that the proposed algorithms yield substantial computational savings without reducing result quality.

Index Terms—Hotspot detection, statistical significance, spatial networks, spatial data mining

1 Introduction

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

1.1 An Illustrative Application Domain: Preventing Pedestrian Fatalities

Urban computing aims to handle the major issues that cities face by using computational techniques. Seven major urban computing categories are urban planning, transportation, environment, energy, social, economy, and public safety and security [1]. Our proposed work focuses on discovering the statistically significant linear hotspots on road networks to address the issues associated with two of these application domains namely transportation planning and public safety and security.

To illustrate the applicability of significant linear hotspot discovery, we focus on the problem of discovering significant concentrations of pedestrian fatalities in a transportation network. According to a recent policy report, more than 47,700 pedestrians were killed in the United States from 2000 to 2009 [2], and more than 688,000 pedestrians were injured over the same time period, which is equivalent to a pedestrian being struck by a vehicle every 7 minutes. Pedestrian fatalities have increased in many places, including 15 of the country's largest metro areas, even as overall traffic deaths have fallen [2].

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 city streets towards higher speed arterials. This has resulted in more than half of fatal pedestrian crashes occurring on these wide, high capacity and high-speed thoroughfares. Typically designed with four or more lanes and high travel speeds, arterials are not built with pedestrians in mind (Fig. 2a). 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]. Fig. 2c shows a map of pedestrian fatalities that occurred on Orange County roads from 2000 to 2009. Transportation planners and engineers need tools to assist them in identifying which frequently used road segments/stretches pose statistically significant levels of risk for pedestrians and consequently should be redesigned. A promising way to effectively discover such road segments/stretches raises from the availability of the large amount of fatality data and

- *X. Tang, E. Eftelioglu, and S. Shekhar are with the Department of Computer Science and Engineering, University of Minnesota, 4-192 Keller Hall, 200 Union St, Minneapolis, MN 55455. E-mail: {xuntang, emre, shekhar}@cs.umn.edu.*
- *D. Oliver is with ESRI, 380 New York Street, Redlands, CA 92373. E-mail: doliver@esri.com.*

Manuscript received 8 Dec. 2015; revised 30 July 2016; accepted 5 Nov. 2016. Date of publication 5 Feb. 2017; date of current version 7 June 2017. Recommended for acceptance by Y. Zheng, C.T. Silva, R. Ghani, and C. Masolo. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TBDATA.2016.2631518

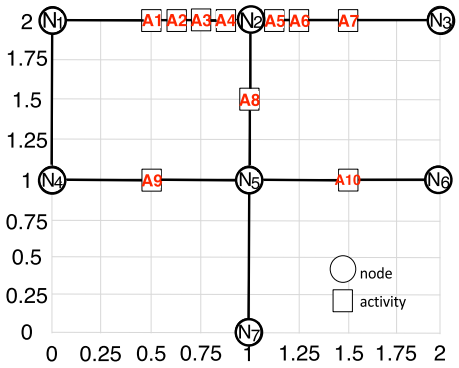


Fig. 1. Example of input of Significant Linear Hotspot Discovery (Best in color).

corresponding data analysis approaches [1]. 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 paper presents an approach for identifying statistically significant linear concentrations of activities such as pedestrian fatalities in a spatial network.

If traditional hotspot discovery is used (e.g., circular hotspot detection), the government can just fix the hotspots for the pedestrian. However, often such hotspots do not occur on a circular area and the fix of those hotspot locations requires a road to be analyzed for structural mistakes by the transportation planners. The Minnesota Department of Transportation proposes that building lane separators for pedestrian sidewalks, bicycles and motorcycles along roads illustrated in Fig. 2b as a way to decrease the risk of traffic accidents [3]. In addition, lack of barriers between opposite traffic on specific parts of highways may lead to severe traffic accidents. For example, since Spring 2015, there were two fatal traffic accidents occurred on Minnesota Highway 280 at similar locations. The cars that lost control went to the flowing traffic on the opposite side since no barriers were in between along the highway segment [5].

Traditional network-based techniques which consider edges atomic face a fundamental limitation in dealing with very long edge with a dense cluster of activities at one end of the edge. These techniques may either fail to capture the precise locations of the hotspots on such edges or even completely miss the hotspots. For example in Fig. 1, $\langle N_1, N_2\rangle$ is an edge whose activities are densely clustered near N_2 (i.e., A_1, A_2, A_3, A_4). Traditional techniques may capture $\langle N_1, N_2\rangle$ as a hotspot, however, the precise location which is actually $\langle A_1, A_4\rangle$ is not captured. In worse case, suppose the concentration of $\langle A_1, A_4\rangle$ exceeds the threshold but the overall concentration of the entire path $\langle N_1, N_2\rangle$ does not since it is compromised by the empty end $\langle N_1, A_1\rangle$, this hotspot will be completely missed. To address this limitation, domain experts introduces *dynamic segmentation* which segments edges into sub-edges at each activity. This paper investigates *dynamic segmentation* in order to evaluate if paths between activities are hotspots. For example, $\langle A_1, A_4\rangle$ will be returned as a hotspot with a more precise location than $\langle N_1, N_2\rangle$ returned by the traditional techniques. In the other case that $\langle N_1, N_2\rangle$ is not qualified as a hotspot, $\langle A_1, A_4\rangle$ may also be return as a hotspot. The technical details of *dynamic segmentation* are elaborated in the problem statement (Section 2.2). The significance of

(a) (b) (c)

Fig. 2. (a) Pedestrian at risk on a road without proper sidewalks [2]. (b) Paved sidewalk and road separators for pedestrians, bicycles, and motorcycles build on road [3]. (c) Pedestrian fatalities occurring on arterials in Orange County, FL [4] (Best in color).

using *dynamic segmentation* is demonstrated in the case studies on real datasets (Section 5).

Linear hotspot discovery can also be applied to other application domains. Certain types of crimes (e.g., assaults, street robbery) may form hotspots on transportation networks that represent roads needing more attention from police departments [4]. In addition, water quality changes on river networks may form hotspots that represent bursts of pollution [6].

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 and road segments. Enumerating and evaluating all shortest paths at the sub-edge level results in prohibitive computational cost. Additionally, density ratio does not obey the monotonicity property, meaning that there is no ordering between the density ratio of a path and its super-paths, or vice-versa. Furthermore, depending on the method used to determine statistical significance, computation times may also be impacted (e.g., $m = 1000$ Monte Carlo simulations may be required to calculate statistical significance).

1.3 Related Work and Their Limitations

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

Geometry-based techniques [7], [8] partition spatial data using geometric shapes (e.g., circles, rectangles). This is useful in domains such as public health, where finding spatial clusters with a higher density of disease is of interest for understanding the distribution and spread of diseases, outbreak detection, etc. Kulldorff, et al. proposed a spatial scan statistics framework (and the SaTScan software) for disease outbreak detection [9]. The spatial scan statistic employs a likelihood ratio test where the null hypothesis is the probability that disease occurrence inside a region is the same as outside, and the alternative hypothesis is that there is a higher probability of disease inside than outside. All the spatial regions, represented by a circle or ellipsoid in the spatial framework, are enumerated and the one that maximizes the likelihood ratio is identified as a candidate. However, if we apply SaTScan to a road network, many significant routes may be missed since a large fraction of the area bounded by circles for activities on a path will be empty. Fig. 3a shows an example of the output of SatScan. Two circular hotspots are detected with large empty areas, which result in high p-values (i.e., 0.125 and 0.497).

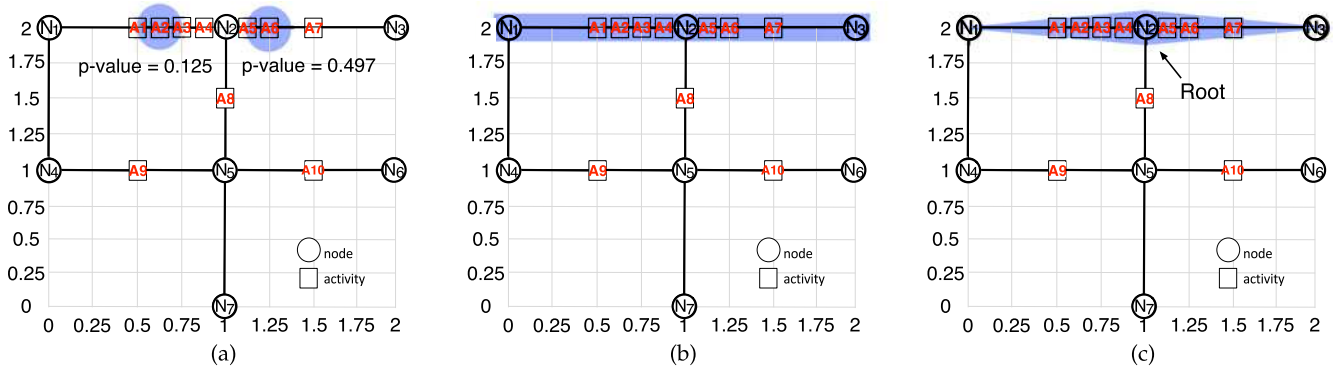


Fig. 3. Example (a) output of SatScan [8], (b) output of linear intersecting paths (LIP) [11], and (c) output of constrained minimum spanning trees (CMST) [10] (best in color).

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.).

By contrast, network-based techniques [10], [11], [12] leverage the underlying spatial network when partitioning spatial data. For example, Linear Intersecting Paths (LIP) [11] and Constrained Minimum Spanning Tree [10] utilize a subgraph (e.g., a path or tree) to discover statistically significant groups.

LIP [11] discovers one anomalous sub-component of a set of connected paths that intersect each other. The connected paths are based on locations in the spatial network with the highest percentage of activities, specified by the user. Hence the density ratio is evaluated only on a portion of the graph specified by this percentage, not on the entire spatial network. Fig. 3b shows an example of the output of LIP. The user-specified percentage is 40 percent, which means all the candidates will have paths containing edge $\langle N_1, N_2 \rangle$ since this edge has 4 activities (out of 10 activities). Examples of possible candidates are $\langle N_1, N_3 \rangle$, $\langle N_1, N_5 \rangle$, $\langle N_2, N_4 \rangle$, $\langle N_1, N_7 \rangle$, etc. The output is $\langle N_1, N_3 \rangle$, since it has the highest density ratio. However, in addition to returning only one statistically significant component, the results of this approach are sensitive to the percentage of the network selected. If the percentage is too high, the number of candidates may be highly restricted, which could result in not identifying statistically significant regions of interest. If the percentage is too low, LIP may be computationally prohibitive due to the large number of candidates.

Another network-based technique, CMST [10], finds one statistically significant tree in the spatial network. Fig. 3c shows an example of the output of CMST. Here the output is $\langle N_1, N_3 \rangle$, where N_2 is the root, since this tree has the highest density ratio. However, this approach also has limitations. In addition to returning only one statistically significant tree, the size of the tree is restricted, which could result in not identifying statistically significant regions of interest.

1.4 Contributions

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

- We propose a new model named dynamic segmentation, which allows the proposed approach to find multiple significant linear hotspots at the sub-edge level in the spatial network. We also introduce new algorithmic refinements to improve the scalability of linear hotspot detection with dynamic segmentation, including a neighbor node filter and a shortest path tree pruning algorithm.
- We analytically prove the correctness of the proposed algorithms and present a cost analysis.
- We present two case studies comparing the detection results under dynamic segmentation with results in the related works, including SRM_GIS [13] and SatScan [8].
- Experimental results on real and synthetic data show that the proposed algorithms, yield substantial computational savings over SRM_GIS [13] without reducing either completeness or correctness of the result.

1.5 Scope and Outline of the Paper

This paper focuses on finding significant discrete activity events (e.g., pedestrian fatalities, crime incidents) associated with a point on a network. This does not imply that all activities must necessarily be associated with a point in a street. In addition, other network properties such as GPS trajectories and traffic densities of road networks [14], [15] are not considered. In this work, it is assumed that the number of activities on the road network is fixed and does not change over time. We do not consider techniques that do not employ statistical significance (e.g., DBScan [16], K-Means [17], KMR [18], and Maximum Subgraph Finding [19]). This paper only enumerates shortest paths rather than all possible paths. This increases the computational tractability since the enumeration space decreases from exponential to polynomial. We choose shortest paths based on the assumption people generally prefer taking shortest paths a destination. Discovering hotspots on shortest paths may find the most possible "dangerous" routes between two locations.

The paper is organized as follows: Section 2 presents the basic concepts and problem statement of Significant Linear

Hotspot Discovery (SLHD). Section 3 presents our preliminary results towards addressing SLHD. Section 4 details our proposed approaches for discovering sub-edge level linear hotspots in a scalable way. Section 5 presents two case studies comparing the proposed significant network-based outputs (i.e., shortest paths) to a significant geometry-based output (e.g., circles) on pedestrian fatality data. The experimental evaluation is covered in Section 6. Section 7 presents a discussion. Section 8 concludes the paper and previews future work.

2 Basic Concepts and Problem Statement

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

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 n in N is associated with a pair of real numbers (x, y) representing the spatial location of the node in euclidean space [20]. Edge set E is a subset of the cross product $N \times N$. Each element $e = (n_i, n_j)$ in E is an edge that joins node n_i to node n_j.*

Fig. 1 shows an example of a spatial network where circles represent nodes and lines represent edges. A road network is an example of a spatial network where nodes represent street intersections and edges represent streets.

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

In transportation planning, an activity may be the location of a pedestrian fatality; in crime analysis, an activity may be the location of a theft. Some of the edges in Fig. 1 are associated with a number of activities (e.g., edge $\langle N_1, N_2 \rangle$ has four activities).

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

For example, in Fig. 1, the activity coverage *inside* path $\langle A_9, N_5, A_8, N_2, A_5, A_6 \rangle$ is 4 whereas the activity coverage *outside* this path is $10 - 4 = 6$.

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

In Fig. 1, the weight *inside* $\langle A_9, N_5, A_8, N_2, A_5, A_6 \rangle$ is 1.75 whereas the weight *outside* this path is $7 - 1.75 = 5.25$.

Definition 5. *The density ratio of path p, $\lambda_p = \frac{a_p \div w_p}{(|A|-a_p) \div (|W|-w_p)}$ [9], [12].*

The density ratio of path p, λ_p is the ratio of the activity density *inside* path p, $\frac{a_p}{w_p}$ to the activity density *outside* p, $\frac{|A|-a_p}{|W|-w_p}$. Fig. 4 lists 3 shortest paths from Fig. 1, namely $\langle A_9, A_6 \rangle$, $\langle A_4, A_{10} \rangle$, and $\langle A_1, A_7 \rangle$. Path $\langle A_9, A_6 \rangle$ contains activities A_9, A_8, A_5 and A_6 and has a weight of 1.75, hence its density is $\frac{4}{1.75}$ while the density outside is $\frac{10-4}{7-1.75}$. Therefore, the density ratio of path $\langle A_9, A_6 \rangle$ is $\frac{4 \div 1.75}{(10-4) \div (7-1.75)} = 2$. By similar calculation, path $\langle A_4, A_{10} \rangle$ has a density ratio of 1.42 and path $\langle A_1, A_7 \rangle$ has a density ratio of 14.

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

Fig. 4. Examples of density ratio.

The reason why we use density ratio as the test statistic in this paper is two-fold. First, density ratio is in a family of metrics inspired from the hypothesis test in which the null hypothesis is "the density inside and outside a path are equal" while the alternative hypothesis is "the density inside a path is larger than outside". These metrics are largely used in hotspot detection literature [7], [9], [21] and follow three properties [7]: (1) Given a fixed weight, the metric increases monotonically with activity coverage. (2) Given a fixed activity coverage, the metric decreases monotonically with weight. (3) Given a fixed ratio of activity coverage to the weight, the metric increases monotonically with the weight. Any metrics that follow these properties (e.g., log likelihood ratio [9]) can be directly applied in the proposed approaches without any algorithmic changes. Second, among these metrics, density ratio is widely used in the literature that deals with activities associated with spatial networks [12]. We use density ratio to make it easier to compare the proposed algorithms with the literature.

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

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

Edges $\langle N_1, N_2 \rangle$ and $\langle N_2, N_3 \rangle$ in Fig. 1 are active edges because they each have at least one activity, and nodes N_1, N_2, N_3, N_5, N_6, and N_7 are all active nodes because they are all joined by active edges. By contrast, Node N_4 is an inactive node because it is not joined by any active edges.

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

For example, in Fig. 1, $\langle N_1, N_2, N_5, N_6 \rangle$ and $\langle N_1, N_2, N_5, N_7 \rangle$ are super-paths of $\langle N_1, N_2, N_5 \rangle$. Conversely, $\langle N_1, N_2, N_5 \rangle$ is a sub-path of $\langle N_1, N_2, N_5, N_6 \rangle$.

2.2 Problem Statement

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

Given:

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

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

Find. All shortest paths $r \in R$ with $\lambda_r \geq \theta_\lambda$, p-value $\leq \theta_p$

Objective: Computational efficiency

Constraints:

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

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

Dynamic Segmentation. Our approach resolves statistically significant routes to the sub-edge level (i.e., routes between activities), which is not investigated in our previous work [13]. This requires a model called dynamic segmentation. Intuitively, it modifies the traditional network structure such that new nodes are formed at the locations of activities and new edges are added to connect these nodes.

The pseudocode in Algorithm 1 shows the process of dynamic segmentation. All the edges with a activities (where $a > 0$) are split into a nodes and $a + 1$ edges (line 2). For clarification, in this model, the nodes before segmentation are referred to as static nodes, while the newly formed nodes, which are essentially activities, are referred to as dynamic nodes. The weights of the newly formed edges in the dynamically segmented network are then updated based on the distance between activities (line 3). In other words, the weight of the dynamic edge formed between activity x and activity y will be updated to the distance between these two activities. Dynamic nodes are stored in $dynamicNodes$, and newly formed edges are stored in $dynamicEdges$ (line 4).

Algorithm 1. Dynamic Segmentation

Input:

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

Output:

A dynamically segmented spatial network G_s derived from G

Algorithm:

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

For example, in Fig. 1, activities A_1, A_2, A_3, etc. become nodes in the spatial network, and edge $\langle N_1, N_2$ is cut into several edges, namely $\langle N_1, A_1\rangle$, $\langle A_1, A_2\rangle$, $\langle A_2, A_3\rangle$, $\langle A_3, A_4\rangle$, and $\langle A_4, N_2\rangle$.

The dynamic segmentation model enables us to evaluate paths that start and end with activities and may be in the middle of an edge. As such, the density ratios tend to be more precise since the extra portions of the path before the first activity and after the last activity are trimmed. Therefore, segments which were previously not tested for statistical significance or which may have been previously deemed "not significant" because they were on a long empty edge, may end up as part of the result. For example, in Fig. 1, $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ and $\langle N_1, N_2, N_3\rangle$ have the same set of activities but the weight of $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ is less. In this case, the density ratio for $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ is much higher (14 versus 5.83), and the p-value is smaller (0.001 versus 0.006).

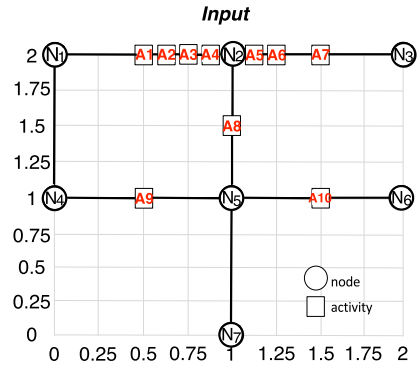
2.2.1 Finding Significant Paths

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

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

3 Preliminary Results

We initially solved the SLHD problem with a previously proposed algorithm (SRM_GIS) [13] featuring two algorithmic refinements: Density Ratio Pruning and Monte Carlo Speedup. Note that even though SRM_GIS was proposed to solve SLHD without dynamic segmentation, it can also be directly applied to dynamically segmented networks. Before describing SRM_GIS, we first review a naïve algorithm Naïve Significant Route Miner(SRM_Naïve) that solves the SLHD problem.



Step 1 | Step 2 | Step 3

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

Output

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

Fig. 5. Example execution trace of Naïve Significant Route Miner. Circles represent nodes and lines represent edges.

3.1 Naïve Significant Route Miner

Algorithm 2 presents the pseudocode for the SRM_Naïve approach. The basic idea behind the algorithm is to find all statistically significant shortest paths in the dynamically segmented spatial network whose density ratio exceeds the threshold θ_λ, under the constraint that the shortest paths returned are not sub-paths of any other path in the output. Algorithm 2 proceeds by calculating all-pairs shortest paths, P, in the spatial network (Line 2). Line 3 evaluates each shortest path in P to determine if it meets the given θ_λ to form a $Candidates$ set. In line 4, the statistical significance of each shortest path in $Candidates$ is evaluated and the significant routes are stored in $SigRoutes$. In order to assess statistical significance, all shortest paths in each of the m simulated graphs are used to calculate the p-value. In line 5, all paths in $SigRoutes$ that are not sub-paths of any other path in $SigRoutes$ are returned, and the algorithm terminates. The purpose of returning significant routes that are not sub-paths of any other path is to improve solution quality. For example, if $\langle N_1, N_2\rangle$ and $\langle N_1, N_2, N_3\rangle$ are both found to be significant, only $\langle N_1, N_2, N_3\rangle$ is returned.

Algorithm 2. Naïve Significant Route Miner Algorithm

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

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

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

SRM_Naïve Example. Fig. 5 shows an example execution trace of SRM_Naïve. The spatial network has seven nodes, seven edges, and 10 activities with specific locations on the edges. The density ratio threshold θ_λ is set to 12, the p-value threshold θ_p is set to 0.001, and the minimum distance threshold θ_ϕ is set to 0.75.

In step 1, all-pairs shortest paths in the given dynamically segmented spatial network are calculated (only paths with high density ratios are shown in the figure). For example, the shortest path between nodes A_1 and A_2 is $\langle A_1, A_2\rangle$. In step 2, the density ratio, λ, for each shortest path is determined (see Definition 5) and paths whose $\lambda \geq \theta_\lambda$ and $\phi \geq \theta_\phi$ are stored as candidates. In the figure, from the six paths listed whose $\lambda \geq \theta_\lambda$, only the first two paths are considered as candidates since their distances meet or exceed the threshold. In step 3, the statistical significance of each candidate is calculated using Monte Carlo simulations (discussed next). Both of the two candidates meet the p-value threshold of 0.01. In step 4 (shown as the output), all paths among the significant paths that are not sub-paths of any other path are returned as significant routes. In this example, path $\langle A_1, A_2, A_3, A_4, N_2, A_5, A_6, A_7\rangle$ are returned since it is the super-path of the other candidate. To reduce the prohibitive computational cost of SRM_Naïve, a new algorithm (SRM_GIS) is proposed in our previous work [13].

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

The SRM_GIS algorithm uses filter and refine techniques (e.g., density ratio pruning and Monte Carlo speedup) to achieve computational savings. The filter and refine techniques may not change worst case complexity but they can reduce runtime in many cases. Density ratio pruning creates a boundary via an upper-bound density ratio such that not all destinations are visited from each source node. Some of the destinations are pruned because the shortest paths to them will never meet the likelihood ratio threshold. Monte Carlo speedup avoids generating all shortest paths in cases where a shortest path in the simulated dataset has a higher density ratio than the shortest paths in the original dataset. Monte Carlo speedup also terminates early if the p-value threshold will not be met based on the number of times the

maximum density ratio in the simulated dataset beats the maximum density ratio in the original dataset.

3.2.1 Density Ratio Pruning

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

The intuition behind the upper-bound density ratio for path p is that (1) the number of activities on all of p's super-paths rooted at p's start node are bounded by the number of activities in the spatial network minus the number of activities in the current shortest path tree rooted at the source node in p and (2) the weight of any super-path of p is at least the weight of the closest edge to p plus p's weight. Using the upper-bound density ratio makes it possible to terminate the all-pairs shortest paths computing earlier. However, since the dynamically segmented network varies during each Monte Carlo simulation trial, the shortest paths need to be computed in each trial. Even though each trial may be early terminated by the upper-bound pruning, the cost is still high. In addition, to ensure the correctness of the results, a post-processing step to verify if the paths with high density ratios are actual shortest path needs to be added to each simulation trial. More details of density ratio pruning are in our previous paper [13]. Due to the limited speedup, density ratio pruning is no longer used in the proposed approaches in this paper.

3.2.2 Monte Carlo Speedup

Monte Carlo speedup aims to calculate the p-value without the need to consider all shortest paths in each simulated graph. The basic idea is that once a shortest path in the simulated graph is found to have a higher density ratio than the maximum density ratio in the original graph, the simulation immediately ends with the p-value being incremented. In other words, there is no reason to keep looking at all shortest paths in the simulated graph if we find one that already beats the maximum density ratio in the original graph. Additionally, Monte Carlo speedup stops all simulations the moment p out of m simulations are found where the simulated density ratio beats the original maximum density ratio. In other words, there is no reason to execute all m simulations if we find that the p-value threshold will not be met.

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

Proof. Please see the extended version of this paper [22]. □

4 Proposed Approach

In Section 3, we reviewed SRM_GIS [13] to address the SLHD problem. However, the high computational cost makes it not capable to handle large amount of data. In this section, we propose a new algorithm (SRM_TBD) to significantly scale up the solution to the SLHD problem. Two new algorithmic refinements, namely neighbor node filter algorithm and shortest path tree pruning algorithm (SPTP) are introduced, respectively.

4.1 Algorithms

4.1.1 Neighbor Node Filter

The idea behind the neighbor node filter is to obtain the shortest path between a pair of activities by stitching together a shortest path between two static nodes in the original network (i.e., the network before dynamic segmentation) with paths between the activities and the static nodes. Recall that in SRM_GIS [13], due to the activity shuffle, the dynamically segmented spatial network changes during each Monte Carlo simulation trial, which requires computing all-pairs shortest paths in each trial. In addition, since activities are considered as dynamic nodes in the dynamically segmented spatial networks, the cost for all-pairs shortest path computing gets higher than it would in the original network. Neighbor node filter reduces the computational cost in two ways. First, since the network does not change during Monte Carlo simulation trials, all-pairs shortest paths only need to be computed once. Second, only shortest paths between static nodes need to be computed. This reduces the all-pairs shortest path computing cost compared with SRM_GIS.

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

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

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

The speedup of the neighbor node filter comes from two directions. First, instead of computing the all-pairs shortest paths on the dynamically segmented graph, it only computes the shortest paths between static nodes. Second, it avoids computing all-pairs shortest paths in each Monte

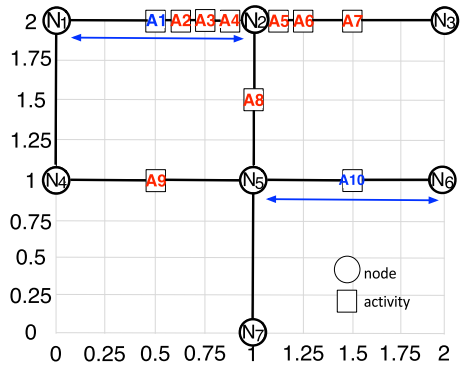


Fig. 6. Example of Neighbor Node Filter for SLHD. Shortest paths between activities are determined by stitching together (1) shortest paths between nodes in the statically segmented network with (2) paths between activities and the start and end of their original edges. Shortest path $\langle A_1, A_{10}\rangle$ is calculated by stitching together $\langle A_1, N_2\rangle$, $\langle N_2, N_5\rangle$, and $\langle N_5, A_{10}\rangle$ (Best in color).

Carlo simulation trial. In Section 4.2 and Section 6, we will evaluate in theoretically and experiments, respectively.

Neighbor Node Filter Example of Step 1. Fig. 6 illustrates an example of the neighbor node filter. Shortest path $\langle A_1, A_{10}\rangle$ is calculated by selecting the shortest among 4 stitched together paths: (1) $\langle A_1, N_1\rangle$, $\langle N_1, N_5\rangle$, $\langle N_5, A_{10}\rangle$; (2) $\langle A_1, N_1\rangle$, $\langle N_1, N_6\rangle$, $\langle N_6, A_{10}\rangle$; (3) $\langle A_1, N_2\rangle$, $\langle N_2, N_5\rangle$, $\langle N_5, A_{10}\rangle$; and (4) $\langle A_1, N_2\rangle$, $\langle N_2, N_6\rangle$, $\langle N_6, A_{10}\rangle$. As the result, shortest path $\langle A_1, A_{10}\rangle$ is $\langle A_1, N_2\rangle$, $\langle N_2, N_5\rangle$, $\langle N_5, A_{10}\rangle$.

4.1.2 Shortest Path Tree Pruning

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

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

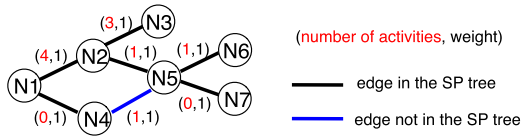


Fig. 7. Example of shortest path tree pruning.

to the subtree rooted at N_{end} except the edges in the path under search. The lower-bound of weight $\underline{w}$ is the weight of path $\langle N_{start}, N_{end}\rangle$ (line 8). After that, the upper-bound of density ratio $\overline{\lambda}$ is computed (line 9) from $\overline{a}$ and $\underline{w}$. If $\overline{\lambda}$ is equal to or exceeds the density ratio threshold θ_λ, all the activities adjacent to N_{end} are put into the list of each activity adjacent to N_{start}, and the search continues on the subtree rooted at N_{end} (line 10–13). Otherwise, the search ignores the subtree rooted at N_{end} which means that any of the activities adjacent or within it is impossible to pair with any activity adjacent to N_{start}. Note that the activities adjacent to the same active node, which means the path between them may not contain a path between static nodes, are always put in the list.

Algorithm 4. Shortest Path Tree Pruning Algorithm

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

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

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

Shortest Path Tree Pruning Example: Fig. 7 shows the shortest path tree rooted at N_1 from the graph in Fig. 1. Note that the blue edge between N_4 and N_5 is an edge in the graph but not in the tree. Each edge is associated with two numbers, indicating its number of activities (red) and its weight (black). We pick the path having the largest number of activities as the shortest path if there are more than one shortest paths of equal weight between a pair of nodes.

With the density ratio threshold set to 12, the algorithm starts from N_1, searching either edge $\langle N_1, N_2\rangle$ or edge $\langle N_1, N_4\rangle$ as the first step. For the purpose of illustration, we select $\langle N_1, N_2\rangle$. Then, we compute the upper-bound of the density ratio $\overline{\lambda}$ of edge $\langle N_1, N_2\rangle$ by computing $\overline{a}$ and $\underline{w}$. To compute $\overline{a}$ of edge $\langle N_1, N_2\rangle$, we add four numbers: (1) the number of activities on $\langle N_1, N_2\rangle$; (2) the number of activities on $\langle N_1, N_4\rangle$ since it has the largest number of activities among all the edges adjacent to N_1 except $\langle N_1, N_2\rangle$; (3) the number of activities on the subtree rooted at N_2; and (4) the number of activities on $\langle N_2, N_3\rangle$ since it has the largest number of activities among all the edges within or adjacent to the subtree rooted at N_2 except $\langle N_1, N_2\rangle$. Hence, $\overline{a} = 4 + 0 + 5 + 3 = 12$. The lower-bound of weight $\underline{w}$ is the weight of $\langle N_1, N_2\rangle$, which is 1, therefore, $\overline{\lambda} = \infty$. Since ∞ exceeds the density ratio threshold (i.e., 12), the search continues and the 8 activities adjacent to N_2 are put into the list of each of the 4 activities adjacent to N_1. Next, The search continues on the path $\langle N_1, N_5\rangle$. To compute $\overline{a}$ of path $\langle N_1, N_5\rangle$, we add four numbers: (1) the number of activities on $\langle N_1, N_5\rangle$; (2) the number of activities on $\langle N_1, N_4\rangle$ since it has the largest number of activities among all the edges adjacent to N_1 except $\langle N_1, N_2\rangle$; (3) the number of activities on the subtree rooted at N_5; and (4) the number of activities on $\langle N_5, N_6\rangle$ since it has the largest number of activities among all the edges within or adjacent to the subtree rooted at N_2 except $\langle N_1, N_5\rangle$. Hence, $\overline{a} = 5 + 0 + 1 + 1 = 7$. The lower-bound of weight $\underline{w}$ is the weight of $\langle N_1, N_5\rangle$, which is 2, therefore, $\overline{\lambda} = 5.83$. Since 5.83 is smaller than the density ratio threshold (i.e., 12), the search on this branch terminates and turns to path $\langle N_1, N_3\rangle$. A list of activity pairs will be found during the depth-first search on the shortest path tree rooted at each active nodes.

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

Algorithm 5. Significant Route Miner with Neighbor Node Filter, Shortest Path Tree Pruning, and Monte Carlo Simulation Speedup (SRM_TBD)

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

4.2 Theoretical Analysis

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

Lemma 1. *The shortest path enumerated in the neighbor node filter is correct.*

Proof. Please see the extended version of this paper [22]. □

Theorem 2. *SRM_NN is complete and correct.*

Proof. Please see the extended version of this paper [22]. □

Lemma 2. *Shortest path tree pruning is correct pruning.*

Proof. Please see the extended version of this paper [22]. □

Theorem 3. *SRM_TBD is complete and correct.*

Proof. Please see the extended version of this paper [22]. □

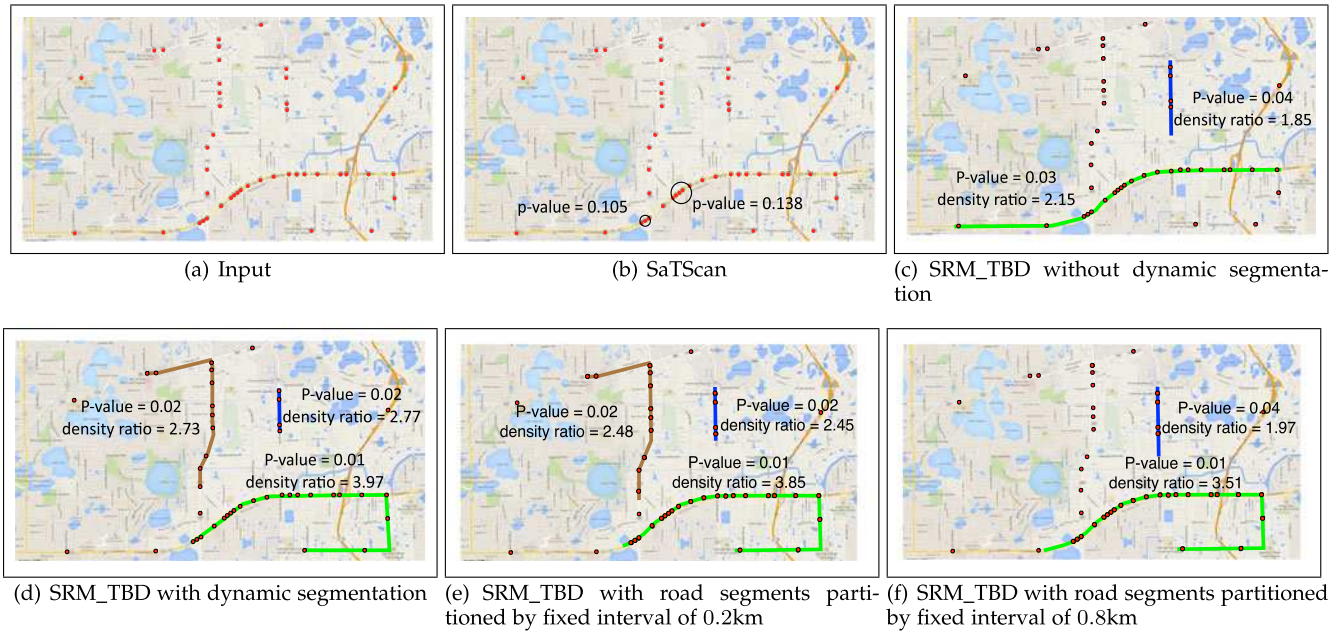
Computational Cost Analysis. The computational costs of SRM_Naïve, SRM_GIS and the proposed algorithms SRM_NN and SRM_TBD stem from 1) the cost of calculating all-pairs shortest path and 2) the cost of assessing statistical significance for all shortest paths in the spatial network.

Cost of SRM_Naïve and SRM_NN. Please see the extended version of this paper [22].

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

We can see that the speedup over SRM_GIS comes from the all-pairs shortest path computing. First, in SRM_NN, all-pairs shortest paths need to be computed only once instead of being computed in each Monte Carlo simulation trial as in SRM_GIS. This decreases the cost a lot since it typically needs $100 \sim 1,000$ trials in the Monte Carlo simulation. In addition, number of nodes taken into consideration decreases from $|N_S| + |N_D|$ to $|N_S|$. This difference can be large since $|N_D|$ could be big when the time period of the dataset is long.

Cost of SRM_TBD. The total cost for SRM_TBD is $O(|N_S|^2 log|N_S| + f_{MC} \times m \times (f_{DFS} \times |N_D|(|N_D| + |N_S|) + f_{SP} \times |N_D|^2))$. Compared with SRM_NN, it aims to reduce the cost of the path evaluation part. In each Monte Carlo simulation trial, depth first searches are run on N_D shortest path trees, each costing $O(|N_D| + |N_S|)$. However, the depth first searches are not necessarily complete with existence of the pruning, therefore, f_{DFS} is multiplied to indicate the speedup from the pruning, where $0 \leq f_{DFS} \leq 1$. Moreover, the path evaluation does not need to go through all activity



(a) Input (b) SaTScan (c) SRM_TBD without dynamic segmentation

(d) SRM_TBD with dynamic segmentation (e) SRM_TBD with road segments partitioned by fixed interval of 0.2km (f) SRM_TBD with road segments partitioned by fixed interval of 0.8km

Fig. 8. Comparing SRM_TBD (without dynamic segmentation), SRM_TBD (without dynamic segmentation, roads are partitioned by fixed intervals), SRM_TBD (with dynamic segmentation), and SaTScan's output on pedestrian fatality data from Orlando, FL [4] (Best in color).

pairs, f_{SP} shows a speedup from that, where $0 \leq f_{SP} \leq 1$. In worst case, SRM_TBD costs more than SRM_NN since the depth first search takes some time but may not prune anything. However, in practice, the pruning ratio is always high, giving SRM_TBD a significant speedup over SRM_NN (shown in Section 6).

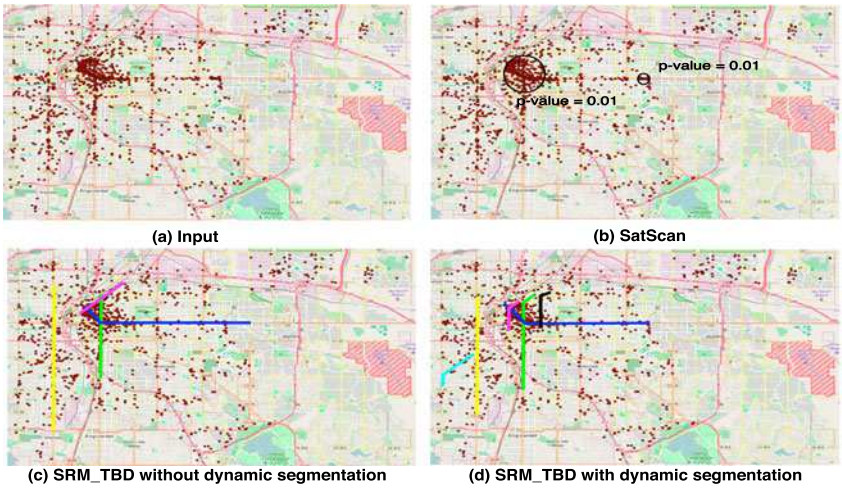
5 Case Study

We conducted two qualitative evaluations of our algorithm (SRM_TBD) with and without dynamic segmentation by comparing their results with the results of SaTScan [23] on two real data sets. First, we used a real pedestrian fatality data set [4], shown in Fig. 8a. As noted earlier, SaTScan discovers areas of significant activity that are represented as circles on the spatial network while SRM_TBD discovers significant shortest paths. The input consisted of 43 pedestrian fatalities (represented as dots) in Orlando, Florida occurring between 2000 and 2009. For each edge (portion of road) in the network, fatality count was aggregated, yielding an overall activity number. Weights of edges were the actual road network distance. The maps were prepared using QGIS' Open Layers plugin [24], and the network was from the US Census Bureau's TIGER/Line Shapefiles [25].

To evaluate the techniques, we considered the outputs of circles versus shortest paths. We used a p-value threshold of 0.04 for our linear hotspot discovery approach. As noted earlier, pedestrian fatalities usually occur on streets, particularly along arterial roadways [2]. Thus this activity can be said to have a linear generator. However, the results with high p-values generated by SaTScan do not capture this. From Fig. 8b, it is clear that SatScan's circle-based output is meant for areas, not streets. In contrast, the shortest paths detected by SRM_TBD without dynamic segmentation fully capture the significant activities on the arterial roads (some of the paths in Fig. 8c are overlapping). Furthermore, the paths in the figure make sense in this context due to the inherently linear nature of the activities.

We further compared SRM_TBD outputs generated with and without dynamic segmentation. With dynamic segmentation (Fig. 8d), SRM_TBD is able to detect the linear hotspot marked as brown (on the road running vertically in the middle of the figure) which is completely missed without dynamic segmentation. The reason is that the overall density ratio of this vertical road is compromised by the empty portion at the bottom, making it not qualified as a hotspot. However, with dynamic segmentation, the dense portion of that road is found as a hotspot while the empty portion at the bottom is dismissed. This contrast shows that dynamic segmentation assists in discovering statistically significant hotspots that are previously missed. In addition, with dynamic segmentation, SRM_TBD gives hotspots with more accurate and precise locations and higher density ratios. For example the blue hotspots in Figs. 8c and 8d indicate the same road. However, the hotspot with dynamic segmentation indicates the precise location where the activities are clustered while the hotspot without dynamic segmentation contains empty portions at both the top and bottom end. Quantitatively, with dynamic segmentation, the density ratio of the blue hotspot is higher (i.e., 2.77 versus 1.85) and the p-value is lower (i.e., 0.02 versus 0.04). These results demonstrate that dynamic segmentation assists in discovering the location of hotspots more precisely than the other technique.

Also, we partitioned long road segments in this network with fixed intervals of 0.2km and 0.8km respectively, and then ran SRM_TBD without dynamic segmentation. The results detected using a small interval of 0.2km shown in Fig. 8e are close to the results detected with dynamic segmentation shown in Fig. 8d with minor difference at the ends of the hotspots, making the density ratios slightly smaller. Fig. 8f shows the results detected using a large interval of 0.8km, in which the vertical hotspot (brown color) is missed. We can see that the results are subject to the length of the interval which is user-specified. If the



P-values and density ratios of results without dynamic segmentation

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

P-values and density ratios of results with dynamic segmentation

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

Fig. 9. Comparing SRM_TBD (without dynamic segmentation), SRM_TBD (with dynamic segmentation), and SaTScan's output on street robbery data from Denver, CO [4] (Best in color).

interval is large, this method may still miss some patterns compared to dynamic segmentation; if the interval is small, this method may have comparable solution quality to dynamic segmentation, however, the computational cost will be expensive since the number of nodes in the partitioned network is much larger than in the original network.

In order to evaluate the SRM_TBD in larger datasets, we also conducted a case study on a dataset of 1,529 simple assaults in Denver, Colorado during 2015 [26] with a p-value threshold of 0.01. As shown in Fig. 9b, SatScan finds a big circular hotspot that covers the downtown area as well as another small hotspot in the middle of the study area. Using a minimum distance threshold of 1km, SRM_TBD without dynamic segmentation discovers 4 nonoverlapped hotspots on the major roads shown in Fig. 9c. SRM_TBD with dynamic segmentation discovers 6 nonoverlapped hotspots shown in Fig. 9d. Some of them (e.g., yellow, green, blue) indicate the same roads as those without dynamic segmentation but capture slightly different portions of the roads, giving higher density ratios. In addition, some previously missed hotspots are captured using dynamic segmentation such as the cyan, pink, and black hotspots. The p-values and density ratios of the results are listed in the right side of Fig. 9. Case study on this larger dataset also demonstrates that dynamic segmentation helps locate the hotspots more precisely and find previously missed hotspots.

6 Experimental Evaluation

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

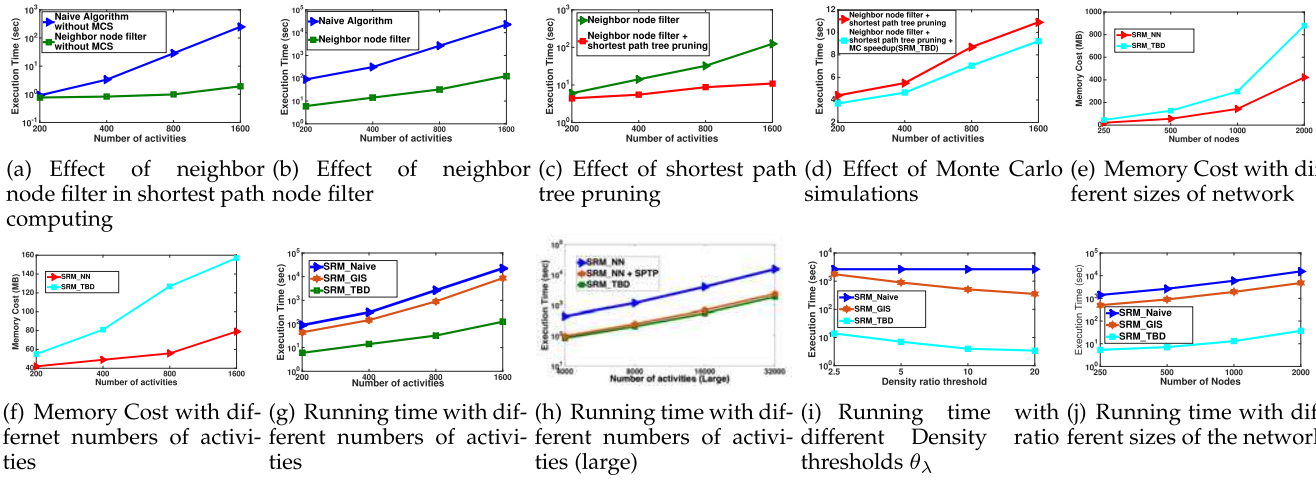
6.1 Experiment Set Up

Our experiments were performed on a real-world road network dataset obtained from the US Census Bureau's TIGER/Line Shapefiles [25] that contained about 500 nodes and 1,000 edges. The weight of each edge was the actual road network distance. The varying number of activities used in the experiments were synthetic data generated under the complete spatial randomness. Network size was varied by putting virtual nodes on the edges. All experiments were performed on a Macbook Pro with an Intel Core i7 Quad Core 2.2 GHz processor and 16 GB RAM.

6.2 Effect of Algorithmic Refinements

Effect of the Neighbor Node Filter. The experiment to evaluate the neighbor node filter had two parts. The first part was designed to test how much speedup is earned without Monte Carlo (MC) simulations. We compared the running time between SRM_Naïve and SRM_Naïve with the neighbor node filter in one run of the all-pair shortest paths computing and density ratio evaluation with varying number of activities. The density ratio threshold was set to 5. Fig. 10a shows the execution times in log scale. We found that the cost of SRM_Naïve grows much larger as the number of activities increases while the cost remains steady with the neighbor node filter. The reason is that the cost of all-pairs shortest paths computing scales up with the number of nodes in the dynamically segmented graph, which increases as the number of activities increases. In contrast, with the neighbor node filter, all-pairs shortest paths are only computed on the original graph, whose size does not vary with the number of activities. The slight increase of cost comes from the density ratio evaluation. The second part of the experiment aimed to evaluate the speedup coming from the MC simulations. The parameters was set the same as the first part except we ran 100 MC simulation trials with a p-value threshold set to 0.05. Fig. 10b shows the execution times in log scale. We found that the speedup of neighbor node filter increases from 10 to 100 times as the number of activities increases. This speedup comes from both the all-pairs shortest paths computing as shown in the first part and the re-use of the shortest paths.

Effect of Shortest Path Tree Pruning. This experiment aimed to evaluate the speedup earned from the shortest path tree pruning approach. We compared the neighbor node filter algorithm to the approach using both neighbor node filter



(a) Effect of neighbor node filter in shortest path computing (b) Effect of neighbor node filter (c) Effect of shortest path tree pruning (d) Effect of Monte Carlo simulations (e) Memory Cost with different sizes of network

(f) Memory Cost with difernet numbers of activities (g) Running time with different numbers of activities (h) Running time with different numbers of activities (large) (i) Running time with different Density ratio thresholds θ_λ (j) Running time with different sizes of the network

Fig. 10. Experimental evaluations of effect of each algorithmic refinement (Figs. 10a, 10b, 10c, 10d, 10e, and 10f) and comparisons among candidate algorithms under different varying parameters (Figs. 10g, 10h, 10i, and 10j).

and SPTP. The density ratio threshold was set to 5, while the p-value threshold was set to 0.05 and the number of MC simulation trials was set to 100. Fig. 10c shows the execution times in log scale. We found that the cost with SPTP grows more slowly as the number of activities increases.

Effect of Monte Carlo Simulation Speedup. We evaluated the speedup earned from the MC simulation speedup by comparing SRM_TBD's performance with and without Monte Carlo simulation speedup. The density ratio threshold was set to 5, while the p-value threshold was set to 0.05. Fig. 10d shows the execution times in linear scale. We found that the speedup provided from MC speedup keeps about 20 percent.

Memory Cost Test. We evaluated the memory cost of SRM_NN and SRM_TBD under different sizes of network and number of activities, respectively. Fig. 10e shows the memory costs with numbers of nodes varied from 250 to 2,000, and Fig. 10f shows the memory costs with numbers of activities varied from 200 to 1,600. We found that SRM_TBD cost memory as around two times as SRM_NN. The main reason is that SRM_NN maintains only the all-pairs shortest paths but SRM_TBD also maintains the shortest path trees. We found that as the size of the network increased, the memory cost for both algorithms increased relatively fast since the total size of paths and trees increased quadratically. As the number of activities increased, the memory cost for both algorithms increased relatively slow since the total size of paths and trees increased linearly caused by the increase of number of active nodes.

6.3 Results of Comparative-Analysis Experiments

Effect of the Number of Activities. This experiment was designed to compare the performance of all approaches with varying numbers of activities among SRM_Naïve, SRM_GIS and SRM_TBD. The density ratio threshold was set to 5, p-value threshold was set to 0.05 and the number of Monte Carlo simulation trials was set to 100. We varied the number of activities from 200 to 1,600. When the number of activities is 1,600, dynamic segmentation increases the network size by adding 1,600 new nodes to the 500 nodes in the original network. Fig. 10g gives the execution times in log scale. SRM_TBD resides apart from SRM_Naïve and SRM_GIS, especially when the number of activities is larger. For example, when number of activities is 1,600, SRM_TBD was about 100 times faster than SRM_GIS.

In addition, in order to test the scalability of the proposed algorithms in larger datasets, we further varied the number of activities from 4,000 to 32,000 on a network containing 2,000 nodes and 2,500 edges. The new network was generated by putting virtual nodes on the edges of the orginal network. We ran only SRM_NN, SRM_NN with shortest path tree pruning and SRM_TBD on these larger datasets since SRM_Naïve and SRM_GIS cost prohibitive time (e.g., SRM_Naïve cost approximate 50,000 seconds for 2,000 activities). The execution times in log scale given Fig. 10h show that the proposed algorithmic refinements maintained the performance on these datasets and SRM_TBD was able to handle larger datasets within reasonable running time.

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

Effect of the Size of the Network. In this experiment, number of activities was set to 800, p-value threshold was set to 0.05, number of MC simulations was set to 100, and density ratio threshold was set to 5. We varied the number of nodes in the network from 250 to 500, while keeping the ratio between number of nodes and edges. This was realized by adding or removing virtual nodes on the existing edges of the real dataset. The results are shown in Fig. 10j. As observed, the cost of SRM_TBD resides apart from SRM_Naïve and SRM_TBD.

In summary, each of the three algorithmic refinements, namely the neighbor node filter, the shortest path tree pruning, and the Monte Carlo speedup, contributes to reduce the computational cost. The comparative-analysis experiments show that SRM_TBD performs much better than the SRM_Naïve approach and SRM_GIS [13] under the varying of different parameters of the dataset.

7 Discussion

Techniques Without Significance Testing. This paper 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 [16], K-Means [17], KMR [18], and Maximum Subgraph Finding [19]. For example, the algorithm from our previous work [18] on summarizing activities using routes may return routes that are not statistically significant. DBScan [16] finds clusters in euclidean space. However, it is sensitive to the parameter selection and may return chance clusters even on a complete spatial randomness dataset. Post-processing the output of these techniques for statistical significance will not guarantee completeness as some of the clusters returned may not be statistically significant. We will explore ways to include statistical significance testing with traditional methods such as K-Means, etc.

Techniques to Reduce Memory Cost. The proposed algorithm needs to store information of all-pairs shortest paths and shortest path trees, which takes a memory cost of $|N_S|^2$ where $|N_S|$ is the number of road intersections in the spatial network. When the size of the spatial network is large to a point, it not feasible to store all the information in memory. To deal with this problem, two techniques which trade off memory cost with running time can be applied. (1) Store the paths and trees in hard drive instead of memory. In this technique, the specific paths and trees are loaded into memory as they are acquired. This reduces the memory cost by sacrificing the running time brought by I/O cost. However, this cost can be mitigated if the information is stored in a well-indexed structure. (2) Use a hierarchical structure. With a hierarchical structure, the spatial network is partitioned into multiple fragments. Only information within each fragment is stored in memory. As an intuitive example, suppose the spatial network is partitioned into p fragments each having $|N_S|/p$ road intersections. The memory cost of storing shortest paths reduces from $|N_S|^2$ to $(|N_S|/p)^2 \times p = |N_S|^2/p$ plus the cost for storing the boundary graph. However, when information across different fragments is acquired, on-time computation is needed which increase the running time. Detailed discussion of hierarchical structure applied in routing algorithms is available in related literature [27].

8 Conclusions and Future Work

This work explored the problem of significant linear hotspot discovery in relation to important application domains such as preventing pedestrian fatalities and crime analysis which are urgent issues in urban area. 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 a neighbor node filter, shortest path tree pruning, and Monte Carlo speedup to enhance its performance and scalability. Two case studies on pedestrian fatality and crime data validate the superiority of our approach over SatScan for detecting statistically significant hotspots of a linear nature. Experimental evaluation using real-world and synthetic data indicated that the algorithmic refinements utilized by our approach yield substantial computational savings without sacrificing result quality.

In future work, we plan to explore other types of data that may not be associated with a point in a street (e.g., aggregated pedestrian fatality data at the zip code level). The present research is centered on finding high concentrations of activities whose counts and locations are deterministic. However, future work is needed to investigate attributes that may not be deterministic such as delay when moving between nodes, capacity constraints, etc. Additionally, estimating p-values via Monte Carlo simulations may be done in different ways. In the current approach we permuted the activities. Alternatively we can permute activity count which may provide a Poisson distribution assumption. In addition, the proposed problem aims to find significant path that are shortest paths. We plan to investigate how to define conceptually more meaningful paths than shortest paths and develop corresponding scalable algorithms. Finally, finding spatio-temporal linear hotspots is also a direction of our future work. Currently, we are using an aggregated number of activities over time. Instead, in the future, we plan to find hotspots that incorporate temporal information [28], [29]. They are potentially applied to find life-cycle and moving trends of hotspots.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under Grant No. 1029711, IIS-1320580, 0940818 and IIS-1218168, USDOD under Grant No. HM0210-13-1-0005, and University of Minnesota via U-Spatial. We would like to thank Kim Koffolt and the members of the University of Minnesota Spatial Computing Research Group for their comments.

References

[1] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 38:1–38:55, 2014.

[2] M. Ernst, M. Lang, and S. Davis, "Dangerous by design: Solving the epidemic of preventable pedestrian deaths," presented at the Transp. America: Surface Transp. Policy Partnership, Washington, DC, USA, 2011.

[3] M. D. of Transportation, "Bicycle and pedestrian safety initiative," (2016). [Online]. Available: http://www.minnesotatzd.org/events/breakfasts/documents/bike_ped_initiatives.pdf

[4] Fatality analysis reporting system (FARS) encyclopedia, National Highway Traffic Safety Administration (NHTSA), (2014). [Online]. Available: ftp://ftp.nhtsa.dot.gov/fars/

[5] A. News, "Minneapolis man killed in crash on I-94 at highway 280," (2016). [Online]. Available: http://kstp.com/news/interstate-94-highway-280-fatal-crash/4094871/

[6] D. Matthews, S. Effler, C. Driscoll, S. O'Donnell, and C. 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 Oceanography*, vol. 53, pp. 743–759, 2008.

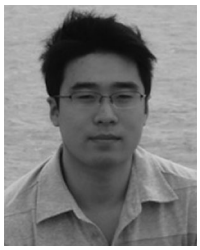
[7] D. B. Neill and A. W. Moore, "Rapid detection of significant spatial clusters," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 256–265.

[8] M. Kulldorff, "Spatial scan statistics: Models, calculations, and applications," in *Scan Statistics and Applications*. Berlin, Germany: Springer, 1999, pp. 303–322.

[9] M. Kulldorff, "A spatial scan statistic," *Commun. Statist.-Theory Methods*, vol. 26, no. 6, pp. 1481–1496, 1997.

[10] M. A. Costa, R. M. Assunção, and M. Kulldorff, "Constrained spanning tree algorithms for irregularly-shaped spatial clustering," *Comput. Statist. Data Anal.*, vol. 56, no. 6, pp. 1771–1783, 2012.

[11] L. Shi and V. P. Janeja, "Anomalous window discovery for linear intersecting paths," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 12, pp. 1857–1871, Dec. 2011.
[12] V. P. Janeja and V. Atluri, "Ls 3: A linear semantic scan statistic technique for detecting anomalous windows," in *Proc. ACM Symp. Appl. Comput.*, 2005, pp. 493–497.
[13] D. Oliver, et al., "Significant linear hotspot discovery: A summary of results," in *Proc. 8th Int. Conf. Geographic Inf. Sci.*, 2014, pp. 284–300.
[14] X. Li, J. Han, J. Lee, and H. Gonzalez, "Traffic density-based discovery of hot routes in road networks," in *Proc. 10th Int. Conf. Advances Spatial Temporal Databases*, 2007, pp. 441–459.
[15] C. Brunsdon, "Computing with spatial trajectories," *Int. J. Geographical Inf. Sci.*, vol. 27, no. 1, pp. 208–209, 2013.
[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery Data*, vol. 96, 1996, pp. 226–231.
[17] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, vol. 1, no. 281–297, Art. no. 14.
[18] D. Oliver, A. Bannur, J. M. Kang, S. Shekhar, and R. Bousselaire, "A K-main routes approach to spatial network activity summarization: A summary of results," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2010, pp. 265–272.
[19] K. Buchin, et al., "Finding the most relevant fragments in networks," *J. Graph Algorithms Appl.*, vol. 14, no. 2, pp. 307–336, 2010.
[20] S. Shekhar and D. Liu, "CCAM: A connectivity-clustered access method for networks and network computations," *IEEE Trans. Knowl. Data Eng.*, vol. 9, no. 1, pp. 102–119, Jan./Feb. 1997.
[21] Y. Zheng, H. Zhang, and Y. Yu, "Detecting collective anomalies from multiple spatio-temporal datasets across different domains," presented at the 23rd Int. Conf. Advances Geographic Inf. Syst., Seattle, USA, 2015.
[22] X. Tang, E. Eftelioglu, D. Oliver, and S. Shekhar, "Technical report: Significant linear hotspot discovery," 2016. [Online]. Available: https://www.cs.umn.edu/research/technical_reports/view/16–037
[23] M. Kulldorff, K. Rand, G. Gherman, G. Williams, and D. DeFrancesco, *SaTScan v 2.1: Software for the Spatial and Space-Time Scan Statistics*. Bethesda, MD: National Cancer Institute, 1998.
[24] Quantum GIS openlayers plugin, (2014). [Online]. Available: http://plugins.qgis.org/plugins/openlayers_plugin/
[25] *Us Census Bureau tiger/line shapefiles*, [Online]. Available: http://www.census.gov/geo/maps-data/data/tiger-line.html, Accessed on: May9, 2014.
[26] Denver crime open dataset, city and county of Denver, (2016). [Online]. Available: https://www.denvergov.org/opendata/dataset/city-and-county-of-denver-crime
[27] S. Shekhar, A. Fetterer, and B. Goyal, "Materialization trade-offs in hierarchical shortest path algorithms," in *Proc. Int. Symp. Spatial Databases*, 1997, pp. 94–111.
[28] P. Mohan, S. Shekhar, J. A. Shine, and J. P. Rogers, "Cascading spatio-temporal pattern discovery," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 11, pp. 1977–1992, Nov. 2012.
[29] X. Zhou, S. Shekhar, and R. Y. Ali, "Spatiotemporal change footprint pattern discovery: An inter-disciplinary survey," *Wiley Interdisciplinary Rev.: Data Mining Knowl. Discovery*, vol. 4, no. 1, pp. 1–23, 2014.



Xun Tang received the bachelor's and master's degrees from Harbin Institute of Technology, Harbin, China, in 2011 and 2013, respectively. He is working toward the PhD degree in computer science at the University of Minnesota-Twin Cities, MN. His research interests include spatial data mining and spatial databases.

Emre Eftelioglu received the bachelor's degree in systems engineering, Turkey, in 2004 and the master's degree in computer engineering from the University of Minnesota, in 2014. He is working toward the PhD degree in computer science at the University of Minnesota-Twin Cities, MN. His research interests include spatial data mining, spatial databases.

Dev Oliver received the bachelor's degree in computer science from Macalester College, in 2004, the master's degree in computer science from the University of Florida in 2008, and the PhD in computer science from the University of Minnesota in 2014. His research and development interests include spatial networks, big data, spatial data mining, spatial databases, and spatial data summarization.

Shashi Shekhar is a McKnight distinguished University professor with the University of Minnesota. For contributions to spatial databases, spatial data mining, and geographic information systems (GIS), he received the IEEE-CS Technical Achievement Award and was elected fellows of the IEEE and the AAAS. He co-authored a text-book on Spatial Databases, and co-edited an Encyclopedia of GIS. He is serving as a member of the Computing Community Consortium Council, a co-editor-in-chief of Geo-Informatica journal, and a program co-chair of G. I. Science Conference (2012). Earlier he served on National Academies committees (Mapping Sciences, GEOINT Research Priorities, and GEOINT Workforce), and editorial board of the *IEEE Trans. on Knowledge and Data Eng.* He also co-chaired Symposium on Spatial and Temporal Databases and the ACM Conference on GIS.