Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 07-025

Discovering and Quantifying Mean Streets: A Summary of Results

Mete Celik, Shashi Shekhar, Betsy George, James P. Rogers, and
James A. Shine

October 25, 2007

Discovering and Quantifying *Mean Streets*: A Summary of Results *

Mete Celik † Shashi Shekhar ‡ Betsy George§ James P. Rogers ¶ James A. Shine ‖

October 12, 2007

Abstract

Mean streets represent those connected subsets of a spatial network whose attribute values are significantly higher than expected. Discovering and quantifying *mean streets* is an important problem with many applications such as detecting high-crime-density streets and high crash roads (or areas) for public safety, detecting urban cancer disease clusters for public health, detecting human activity patterns in asymmetric warfare scenarios, and detecting urban activity centers for consumer applications. However, discovering and quantifying *mean streets* in large spatial networks is computationally very expensive due to the difficulty of characterizing and enumerating the population of streets to define a norm or expected activity level. Previous work either focuses on statistical rigor at the cost of computational exorbitance, or concentrates on computational efficiency without addressing any statistical interpretation of algorithms. In contrast, this paper explores computationally efficient algorithms for use on statistically interpretable results. We describe alternative ways of defining and efficiently enumerating instances of subgraph families such as paths. We also use statistical models such as the Poisson distribution and the sum of independent Poisson distributions to provide interpretations for results. We define the problem of discovering and quantifying *mean streets* and propose a novel *mean streets* mining algorithm. Experimental evaluations using synthetic and real-world datasets show that the proposed method is computationally more efficient than naïve alternatives.

1 Introduction.

Mean streets represent those connected subsets of a spatial network whose aggregated attribute values are significantly higher than expected. Formally, given a road network $G = (V, E)$ and a set of aggregated crime values on edges E, a *mean street* mining algorithm aims to discover and quantify correct and complete sets of connected subsets of the road network. For example, Figure 1 shows " *mean streets*" of a part of a metropolitan city in the United States. Each line represents a street and thickess of it represents the aggregated crime value of the street. In the Figure, the thicker the street is, the higher the crime density is. The *mean street* mining algorithm discovers and quantifies high-crime-density streets in the dataset. (An important note: In this paper, the "mean" in "*mean streets*" refers not to any statistical measure but to the idea of angry or dangerous as in the 1953 film by the same name. To avoid confusion, we use the term "average" in the paper to denote the statistical mean.)

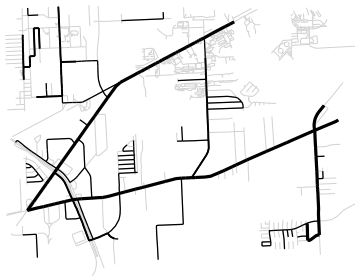


Figure 1: *Mean streets* of a metropolitan city of United States

Discovering and quantifying *mean streets* is very important for many application domains, including crime analysis (high-crime-density street discovery) and police work (planning effective and efficient patrolling strategies). In urban areas, many human activities are centered about spatio-temporal (ST) infrastructure networks, such as transportation, oil/gas pipelines, and utilities (e.g., water, electricity, telephone). Thus, activity reports such as crime reports may often use network based location references (e.g., street addresses). In addition, spatial interaction among activities at nearby locations may be constrained by network connectivity and network distances (e.g., shortest paths along roads or train networks) rather than the geometric distances used in traditional spatial analysis. Crime prevention may focus on identifying subsets of ST networks with

*This work was partially supported by the US Army Corps of Engineers under contract number W9132V-06-C-0011, the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under contract number DAAD19-01-2-0014, and the NSF grant ISS-0431141.

†Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, {mcelik}@cs.umn.edu.

‡Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, {shekhar}@cs.umn.edu.

§Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, {bgeorge}@cs.umn.edu.

¶U.S. Army ERDC, Topographic Engineering Center, VA, USA, {james.p.rogers.II}@erdc.usace.army.mil.

‖U.S. Army ERDC, Topographic Engineering Center, VA, USA, {james.a.shine}@erdc.usace.army.mil.

high activity levels, understanding underlying causes in terms of ST network properties, and designing ST network control policies.

However, identifying and quantifying *mean streets* is challenging for several reasons. One large challenge is choosing the correct statistical model. Many existing ST models have assumptions of data normality and either spatial and temporal homogeneity or a well-defined autocorrelation in these domains. A major limitation is the inadequacy of descriptive and explanatory models for activity around ST networks such as train and road networks. Another challenge is that the discovery process of *mean streets* in large spatial networks is computationally very expensive due to the difficulty of characterizing and enumerating the population of streets to define a normal or expected activity level.

Public safety professionals may be interested in analyzing the ST network factors to explain high activity levels or changes in activity levels at certain highway segments, or to compare prevention options such as check points. Such analysis is not only hard using existing methods, but it may not be statistically meaningful, since common methods such as spatial regression do not adequately model ST network constraints such as connectivity and directions. Thus, we explore a novel ST network analysis method to study descriptive and explanatory models for ST network patterns.

For purposes of this paper, we evaluate graphical models in statistics for their ability to model activities on road networks. Road segments will be modeled as edges or as nodes in graphical models, and similarities and differences in crime rates will be examined. We are particularly looking to find areas where the aggregated crime rate is high, the so-called *mean streets*. We explore computationally efficient algorithms for statistically interpretable results. We describe alternative ways of defining and efficiently enumerating instances of subgraph families such as paths. We also provide statistical models such as the Poisson distribution and the sum of independent Poisson distributions to provide statistical interpretations for results. We define the problem of discovering and quantifying *mean streets*. We propose a novel *mean street* mining algorithm. Experimental results show that the proposed method is computationally more efficient than naïve alternatives.

1.1 Related Work. Previous studies on discovering high-density regions (i.e. hotspots) can be classified into two main categories based on their statistical interpretability. Li et al. defined the hot routes discovery problem in road networks using moving object trajectories [18]. However, discovered patterns in this approach do not have a statistical interpretation such as statistical significance. In addition, this algorithm is designed to process tracks (e.g., GPS tracks) rather than point or aggregate datasets referencing street networks. In contrast, we propose to identify statistics-based methods to identify hotspots.

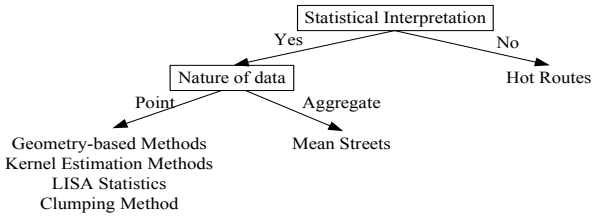


Figure 2: Classification of the related work

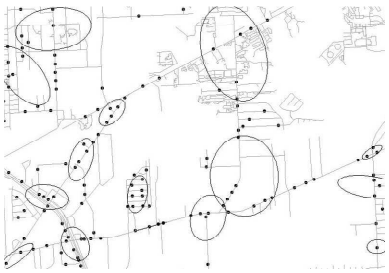


Figure 3: Point data and output of K-means clustering

Statistics-based methods to identify hotspots can be classified into two categories based on the nature of the dataset: point-based methods [1, 6, 11, 17, 25, 27, 29, 22] and aggregate-based methods. Our problem belongs to the latter one. The aim of the point-based approaches is to discover high-density regions from point datasets which show the actual locations of the crimes (Figure 3). The point-based approaches focus on the discovery of the geometry (e.g. circle, ellipse, etc.) of the high-density regions [6]. The Spatial and Temporal Analysis of Crime (STAC) tool in the Crimestat software, nearest neighbor hierarchical clustering techniques, and K-means clustering techniques are among the methods that use the ellipse method to identify hotspots [17]. Figure 3 shows the result of CrimeStat using K-eans clustering method for 15 clusters [17]. Kernel estimation methods have been developed to identify isodensity hotspot surfaces because hotspots may not have crisp ellipsoid boundaries. Local indicators of spatial association (LISA) statistics were proposed to eliminate the limitations of ellipsoid-based and kernel-based estimation techniques [1, 11]. The clumping method was proposed by Roach to discover clumped points (e.g. hotspots) from a point dataset [27]. However, these approaches will not be able to discover and quantify high-crime-density regions (e.g. streets) for given aggregate

crime data. They also do not consider the spatial network structure of the urban dataset, and may not model graph properties such as one-way streets or connectivity. For example, if all crime events occur along a street of a city, these approaches may tend to divide the street into several ellipsoid clusters or may tend to discover a big ellipse where most of the inside of the area has no activity. Shiode and Okabe extended the clumping method for analyzing point patterns on a spatial network [27, 29, 22]. In their approach, if crime point locations on an edge are close enough, they form a clump. A user-defined distance threshold (or clump radius) is used to check if the points are close enough or not. However, their approach will not be able to discover and quantify patterns for aggregate crime data.

Overall, point-based approaches mainly focus on discovering and quantifying hotspots using point crime data. Due to the type of the crime or for victim security, crime location information may not be released by the authorities and aggregated crime values may be released for spatial regions, e.g. streets. In that case, point-based approaches, whether considering the spatial network structure or not, will fail to discover and quantify hotspots when the aggregated crime values are given since these approaches are dependent on the location of the crimes. In contrast, we propose statistics-based methods to discover hotspots (e.g. *mean streets*) from aggregated datasets referencing urban street networks and taking graph semantics into account.

1.2 Contributions. In this paper, we define a problem called *mean streets* based on crime data on a spatial network. We take into account graph properties such as connectivity and edge directionality while discovering and quantifying the *mean streets*. The discovery process is based on a statistical framework. We use a Poisson distribution to model crime incidences. We propose novel and computationally efficient methods for the discovery of *mean streets*. We prove the correctness and the completeness of the proposed methods. We also evaluate the proposed algorithms experimentally using real and synthetic datasets.

1.3 Scope and Outline. The purpose of this paper is to explore and quantify *mean streets* based on aggregated crime incidences. It shows an original, innovative approach for identifying *mean streets*. However, this paper does not take into account the possible errors in crime locations that might occur during the recording.

The rest of the paper is organized as follows. Section 2 presents basic concepts to provide a formal model of a *mean street* and the problem statement of identifying *mean streets*. Section 3 presents our proposed algorithms. Analysis of the algorithms is given in Section 4. Section 5 presents the experimental evaluation with real-world and synthetic datasets and Section 6 presents conclusions and future work.

2 Basic Concepts and Problem Definition.

The focus of this study is to identify and quantify *mean streets* over a spatial network whose aggregated attribute values are higher than a threshold. First we define basic concepts and then we explain how we model *mean streets*.

2.1 Basic Road Network Concepts. A spatial graph $G = (V, E)$ is a set of vertices (e.g. street intersections) connected by edges. Based on the application domain, vertices and edges might have weights (e.g. number of crimes). Graphs can be undirected or directed depending on whether the edges are ordered or not. The lengths d_i of edges may be the same or different. Events (in our case, crimes) $c_1, c_2, ..., c_n$ are distributed across the edges and vertices, and may also be distributed over different time intervals, although we will not consider the time dimension in this paper. The weight on each edge represents the aggregated crime values on the street segment represented by the edge.

2.2 Binomial and Poisson Distributions. If we assume that crimes are distributed uniformly over all edges E, the probability of a single event, e.g. crime c_i, being on a specific edge e_i is the length d_i of that edge divided by the total length of all edges.

$$p = \frac{d_i}{\sum_{i=0}^{m} d_i} \tag{2.1}$$

One option for modeling the distribution of crimes on edges is the binomial distribution [23]. If we call the probability of any given point being on a specific edge e_i, then the probability of k of the n points being on that specific edge can be expressed by [2]

$$Pr(k) = \binom{n}{k} p^k (1-p)^{(n-k)} \tag{2.2}$$

The binomial distribution assumes that the points are independent. A bar plot of the probability mass function of a binomial distribution is shown in Figure 4. The binomial distribution is one of the oldest statistical distributions and has special use in the science of probability and games [21]. Okabe et.al. used this approach as one of several recommended for statistical analysis of points on a network [23].

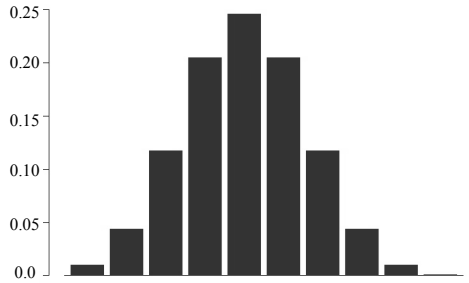


Figure 4: Binomial distribution

Another alternative to the binomial distribution model is the Poisson distribution model. Indeed, when the probability of a given event is small (less than 0.1), the Poisson distribution is a common approximation to the binomial distribution [14]. If we assume 10 or more edges of roughly equal length, the probability of a given crime event falling on a given edge is indeed less than 0.1 and the approximation is valid. In addition, the Poisson computations are more efficient than the binomial computations, although neither would be a large computational issue. For this paper we have chosen to use the Poisson distribution model.

A Poisson distribution can be described by

$$P(x \leq k) = \sum_{i=0}^{k} \frac{\lambda^i \times e^{-\lambda}}{i!} \quad (0 \leq k < \infty) \tag{2.3}$$

where k is the number of occurrences of an event and λ is the average of number of occurrences. This distribution was developed historically to model arrival rates where times are exponential [7]. In recent statistics, it has been used to count the number of rare events, where most data points (or point patterns in spatial and ST applications) do not have an event [26]. One common application for the Poisson distribution is crime analysis [3]. Another usage is for insurance and actuarial claims [13].

The sum of two Poisson processes with parameters λ_1 and λ_2 is also a Poisson process with parameter $(\lambda_1 + \lambda_2)$. If we assume independence of crime rates on different adjacent edges, e.g. streets, (which we do for our model for this paper), this property allows mixing neighboring streets with little computational difficulty [20]. This also allows streets of different lengths to be mixed easily, as the Poisson parameter can be adjusted for differing street lengths and neighboring results can then be added.

One issue which must be at least mentioned with the Poisson distribution is that of unit lengths. For the Poisson distribution, choosing different unit lengths can affect the distribution, mean, and upper level thresholds of the distribution (which are important in Sections 3 and 4). For example, assume a model is fit for an estimated unit length A, and the estimated λ is 3. The .90 and .95 quantiles are 5 and 6 respectively. If we now make the unit length 5A, and the event density is the same, the new estimated λ will be 15. We now would wish the .90 and .95 quantiles to be 5x5=25 and 5x6=30 respectively, but they are actually 20 and 22. Thus the choice of the unit length can significantly affect the discovery threshold for mean streets. One possibility would be to model different categories of streets separately (short, medium and long for example). Another idea is to try a wider range of upper thresholds and see which streets stand out as different the most times. However, if we assume streets that are not markedly different from each other (such as city blocks) we can assume that the difference is not significant, and for the purposes of the paper we have made this assumption. A closer look at this issue would definitely be a topic for future work.

2.3 Modeling *Mean Streets*.

In this section, we provide definitions to model *mean streets*.

Definition 2.1. *Given a road network $G = (V, E)$ and a set of aggregated crime values of edges $E = [e_0, e_1, ...e_m]$, a graph density parameter λ_{est} is formalized as the number of crimes per unit length.*

$$\lambda_{est} = \frac{\sum_i^E crime(e_i)}{length(road\_network)} \tag{2.4}$$

Definition 2.2. *Given a graph density parameter λ_{est}, a confidence threshold α (e.g. percentile threshold) for a Poisson distribution, "mean street" density threshold $\theta(\lambda_{est}, \alpha)$ is the number of crimes per unit length such that a street segment of unit length with $\theta(\lambda_{est}, \alpha)$ crimes falls within the top α percentile of all unit length streets [7]. Formally,*

$$\alpha = P(x \geq \theta(\lambda_{est}, \alpha)) = \sum_{i=\theta}^{\infty} \frac{\lambda_{est}^i \times e^{-\lambda_{est}}}{i!} \tag{2.5}$$

For example, in Figure 5 the graph shows the cumulative distribution function for several estimated parameters λ_{est} (1, 3, 5, and 10). If the user-defined confidence parameter α is 0.9 (which is shown by a dashed line in the graph) and the estimated parameter λ_{est} is 10, the density threshold $\theta(\lambda_{est} = 10, \alpha = 0.9)$ will correspond to the intersection of the cumulative

distribution function and confidence threshold which is close to 14 crimes per unit length.

There are two ways to find the value of $\theta(\lambda_{est}, \alpha)$:
1. Find a closed form formulation of equation 2.5.
2. Calculate the value of the equation 2.5 by incrementing values of $\theta(\lambda_{est}, \alpha)$ until the cumulative probability result is equal to α.

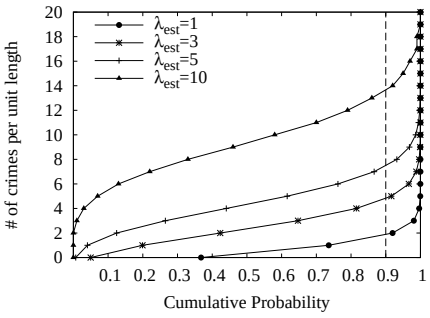


Figure 5: Poisson cumulative distribution function

Definition 2.3. *Given a mean street density threshold $\theta(\lambda_{est}, \alpha)$ and an edge e_i with length d_i, the mean street crime threshold $C_{treshold}$ of edge e_i is the density threshold $\theta(\lambda_{est}, \alpha)$ times length d_i, such that,*

$$C_{treshold} = \theta(\lambda_{est}, \alpha) \times d_i \tag{2.6}$$

Definition 2.4. *Given a road network $G = (V, E)$ with aggregated crime values on the edges E and a density threshold $\theta(\lambda_{est}, \alpha)$, a "mean street" is a connected subset (e.g streets) of the road network whose number of crimes C_{real} is not less than its crime threshold $C_{treshold}$.*

In the next section, we discuss how all paths can be enumerated in a graph. A preliminary step in the discovery of *mean streets* could be the enumeration of all paths in the graph. This can be followed by a discovery of *mean streets* from these paths based on a threshold (Section 3.2).

2.4 Defining and Enumerating Street Populations. Since *mean streets* try to find the connected parts of the road network where the aggregated crime values are significantly higher than the crime thresholds, the first stage in this process could be the enumeration of well-defined subsets of the graph such as connected subgraphs, paths, cycles, and trees. Literature on enumerating subgraphs is most mature in the area of path enumeration and hence the proposed work uses this as the initial stage in finding the *mean streets*. However, this section provides a brief description of enumeration of various types of subgraphs (connected subgraphs, paths, cycles).

2.4.1 Enumeration of All Paths. Here all possible paths in a directed graph are enumerated. The algorithms in this category can be broadly classified into two categories:(1)Algorithms using the powers of an adjacency matrix, and (2)Algorithms using backtracking.

Algorithms based on an adjacency matrix: Initial work in this area dealt with the determination of the number of paths from any given vertex to another [24, 15]. One method was based on a recurrence relation relating paths of length n to paths of length $n-1$ [24]. The matrix of $n-$paths of a graph is expressed in terms of the matrices of $(n-1)$-paths of its first order subgraphs. (A first order subgraph of a graph G is the graph obtained by removing all the edges incident at a vertex u.) The method is based on the result that the matrix of n-paths of a graph G is completely determined in terms of the matrices of $(n-1)$-paths of its first order subgraphs.

An adjacency matrix and its powers have been used to enumerate paths in graphs that contained self-loops and parallel edges [15]. The algorithm counts the number of paths from one vertex to another.

An algorithm that enumerates and lists all paths in a graph was proposed by Rubin [28]. A simple path that connects two vertices would have at most one occurrence of a vertex, thus limiting the maximum number of vertices on any path to n, where n is the number of vertices in the graph. The algorithm uses bit vectors to store the vertices and edges on a path and successively finds paths of increasing lengths using boolean matrix operations. Each entry in the matrix is a list of descriptors and each descriptor has a vertex vector and an edge vector, both boolean. The *vertex vector* has 1's in the positions of vertices that are present in the path. The *edge vector* has 1's in the positions of edges that are present in the path.

Algorithms using backtracking: Backtracking algorithms (for example, [8]) use recursion to find all paths with suitable techniques to avoid infinite traversals of cycles. These algorithms try to find n-paths from a vertex u to a vertex v by enumerating all paths of length $n-1$ from every adjacent vertex w of u to v and appending uv to them.

2.4.2 Enumeration of Disjoint Paths. Given a pair of vertices, finding a pair of paths that do not share edges (edge disjoint paths) has been studied. In a directed graph, determining whether such paths exist

is an NP-hard problem [4]. Approximation algorithms have been proposed to find such paths. Constrained versions of this problem (such as shortest pairs and partially disjoint pairs) have been studied.

Shortest pairs of disjoint paths consist of two edge-disjoint paths of minimum total length from a given source to all other vertices [31]. For a single destination, the problem is solved using a special case of minimum cost network flow in conjunction with Dijkstra's single source shortest path algorithm. This has been extended to all destination nodes.

Algorithms that compute a set of disjoint paths from a source and a destination are available [33]. The paths are found successively starting with the shortest path, increasing the length by an edge at every iteration and removing the edges in the discovered paths from the list of candidate edges. The algorithm assumes uniform weights on edges (=1).

2.4.3 Enumeration of Directed Cycles. The enumeration of all cycles also falls under the categories of (1) Algorithms using the powers of an adjacency matrix (since no cycle can have of length greater than the number of vertices in the graph, the cycles can be computed from the powers of the adjacency matrix (power $\leq$ n) [15, 19].) and (2) Search algorithms (these algorithms search for cycles in an appropriate search space which is a super set of all paths [32].)

Although paths in a graph can be enumerated based on different requirements (e.g., all shortest paths, all cycles, pairwise-disjoint shortest paths), in the crime analysis domain, emuneration of all paths could be the best startegy due to the following reasons: (1) One class of enumeration algorithm which finds disjoint paths between pairs of nodes requires some nodes to be identified as "source" and "destination" nodes. This requirement is not always met in the application domain that is explored in the paper. (2) Although "all pair shortest paths" and cycle enumeration does not require designated node pairs, it is unlikely that the result would contain all required road segments, thus providing the pruning phase with an incomplete search space. Therefore, enumeration of all paths in a graph is likely the best strategy since it assures completeness as required by the *mean streets* discovery process.

2.5 Enumeration of Disjoint Paths. The problem of finding *mean streets* can also be approached by finding all connected subgraphs (not necessarily paths) from the given road network. Enumerating all possible connected subgraphs in a given graph would generate a search space of too enormous a size for the next stages to handle. The partitioning that seems closest to the crime analysis domain would be the one that generates a set of edge-disjoint subgraphs. Appropriate constraints could be added to ensure spatial proximity among the vertices of the subgraphs.

Decomposing graphs into subgraphs that provide an edge cover has been studied [12, 16, 30]. The edges of the graph are covered with a set of edge-disjoint subgraphs. These methods partition the edges into subgraphs satisfying some constraints. For example, in one case [12], the graph is divided into a set of subgraphs such that the total weight of the edges in each subgraph is at most a constant (specified by the user). This problem is NP-complete and approximate algorithm has been proposed [12]. Exact methods and heuristics have also been proposed [16, 30]. These algorithms can be explored to see whether they can be used to find subgraphs that ensure spatial proximity among the nodes.

2.6 Problem Definition. We focus on discovering and quantifying complete and correct sets of *mean streets* for a given road network, an average crime λ_{est}, and a user-defined confidence threshold α. The formal problem definition is as follows.

Given:

- A spatial graph $G = \{V, E\}$, where G is a spatial framework consisting of locations $v_1, v_2, \ldots, v_n$ and E is a collection of edges between locations in V.
- An aggregated attribute function $f : E \rightarrow$ *a set of real numbers* .
- A user-defined confidence threshold α.

Find:

- A set MS of *mean streets* $H = \{e_i | e_i \in E, e_i$ *is a mean street*$\}$ whose aggregated attribute (e.g. crime) values falls within the top α percentile.

Objective:

- Minimize computation cost while finding correct and complete sets of *mean streets*.

Constraints:

- Attribute value of edges in E is a Poisson distribution.

3 Discovering and Quantifying *Mean Streets*.

In this section, we present two novel *mean street* discovery algorithms and then we give an execution trace of the algorithms in Section 3.3. In section 3.1 we discuss an Apriori-based approach to discover and quantify *mean streets*. The idea is to discover *size* $k + 1$

mean streets using *size k mean streets*. This approach has two pruning strategies: i) to eliminate unconnected edge combinations, and ii) to eliminate edge combinations that do not satisfy the crime thresholds. In section 3.2, we discuss a Graph-based approach. The idea is to generate all possible street sets in a spatial network using path generation algorithms and prune the streets that do not satisfy the criteria.

3.1 Apriori-based *Mean Street* Mining Approach. This approach will generate *size* $k+1$ streets using *size* k "*mean streets*" until there are no more candidate streets (Algorithm 1). The inputs of the algorithm are a road network $G = (V, E)$, a set of aggregated crime values C_{real}, and a user-defined confidence threshold α. The output is connected sets of streets whose aggregated crime values is no less than their crime thresholds $C_{treshold}$.

Algorithm 1 Apriori-based *Mean Streets* Mining Algorithm

Inputs:
α: user-defined percentile
$G = (V, E)$: street network
C_{real}: a set of aggregated crimes values
Output: *Mean streets* whose number of crime incidents satisfies crime threshold.
Algorithm:
1: calculate λ_{est}=total_crimes/total_road_length for $G = (V, E)$
2: calculate θ=Poisson (λ_{est}, α)
3: street_size $k = 1$, cand_streets(k)=E, $M\_streets$=*empty*
4: **while** cand_streets not empty **do**
5: Connected(k)=Prune_unconnected_candidates(cand_streets(k), $G = (V, E)$)
6: calculate_crimes_thresholds(θ,Connected(k))
$C_{treshold} = \theta \times length(street)$
7: $M\_streets(k)$=Find_mean_streets whose $C_{real} \geq C_{treshold}$
8: cand_streets(k+1)=generate_candidates(E,$M\_streets(k)$)
9: $k = k + 1$
10: **end while**
11: return union M_street_1, M_street_2, ... M_street_k

In the algorithm, steps 1 and 2 include initialization of the parameters, and steps 5 through 9 give an iterative process to mine *mean streets*. Finally, step 11 gives a union of the results. The functions of the algorithm are explained below.

Calculate the value of λ_{est} (step 1): Parameter λ_{est} is the average number of crimes, e.g., number of crimes per unit length in the spatial network (equation 2.4). The value of the λ_{est} can either be calculated from the given spatial network or supplied by the user.

Calculate the value of $\theta(\lambda_{est}, \alpha)$ (step 2): The parameter $\theta(\lambda_{est}, \alpha)$ is the number of crimes per unit length such that a street segment of unit length with $\theta(\lambda_{est}, \alpha)$ crimes falls within top α percentile of all unit length streets (equation 2.5).

Prune unconnected candidates (step 5): In this step, unconnected candidates are pruned when $k \geq 2$.

Calculate crime thresholds (step 6): The crime threshold of a street is calculated as the product of the length of the street and the parameter $\theta(\lambda_{est}, \alpha)$.

Discover and quantify *mean streets* (step 7): The connected streets whose aggregated crimes are not more than their crime thresholds are pruned in this step. The remaining *size k mean streets* will be used to generate *size* $k+1$ candidate street sets.

Generation of candidate *mean streets* (step 8): This function uses an apriori-based approach to generate *size* $k+1$ candidate streets.

In steps 5-9, the algorithm finds *size* $k+1$ *mean streets*. The algorithm will run iteratively until there are no more candidate *mean streets* to be generated and outputs the union of all size *mean streets*.

3.2 Graph-based *Mean Streets* Mining Approach In this section, we discuss a Graph-based *mean streets* mining approach. Road networks are often represented as graphs and one method to generate *mean streets* is to find all possible paths in the graph (Algorithm 2) and then use an appropriate filtering technique to eliminate the connected street sets that are irrelevant (Algorithm 3). The constraints that need to be satisfied while computing street sets would depend on the users' preferences. For example, in some scenarios, it might be required to generate connected street sets that traverse every edge in the graph at least once. It is also possible that some locations in the road network are designated as start points and end points and the connected street set generation needs to incorporate this requirement.

Algorithm 2 Enumeration of All Simple Paths

```
1: Function EnumeratePaths(Graph G(V, E))
   {Initialization:}
2: for i = 1, n do
3:    for j = 1, n do
4:       D[i, j] = A[i, j]
5:    end for
6: end for
   {Path Computation:}
7: for j = 1, n do
8:    for i = 1, n do
9:       for k = 1, n do
10:         for each entry P_m in D[i, j] and P_l in D[j, k]
11:         if end vertex of P_m = start vertex of P_l then
12:            D[i, k] = append(P_m, P_l);
13:         end if
14:      end for
15:   end for
16: end for
```

Algorithm 2 enumerates all simple paths in a graph. The graph G is represented as an adjacency matrix A, where each entry $A[i, j]$ stores the length of edge (i, j), denoted as d_{ij} or ∞ if edge (ij) is not present in the graph. The algorithm stores the paths computed at every iteration in a matrix D. Each entry in D is a list of pairs. The matrix D is initialized to include all

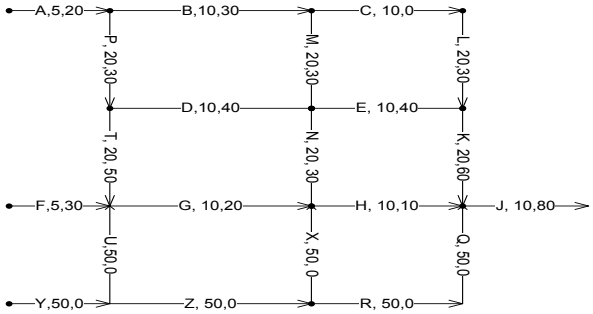
single edge paths (the entries in the adjacency matrix A). Each pair (v_l, e_l) corresponds to a path that has already been computed. The entry v_l lists the vertices in the path and e_l is a list of edges on the path. Both are stored as bit vectors. At each iteration the algorithm checks if a new path can be found by appending two existing paths (Step 11), and if possible, appends the paths and adds the new path to a list in D (Step 12).

The pseudo-code of the Graph-based approach is given in Algorithm 3. In this algorithm, all possible connected street sets are generated in step 1 using Algorithm 2. The rest of the functions of Algorithm 3 are same as described in Section 3.1.

Algorithm 3 Graph-based *Mean Street* Mining Algorithm

Inputs:
α: user-defined percentile
$G = (V, E)$: street network
C_{real}: a set of aggregated crimes values
Output: *Mean streets* whose number of crimes incidents satisfies their crime thresholds.
Algorithm:

1: cand_streets=EnumeratePaths (Graph($G = (V, E)$))
2: calculate λ_{est}=total_crimes/total_road_length for $G = (V, E)$
3: calculate θ=Poisson (α, λ_{est})
4: street_size $k = 1$, $M\_streets$=*empty*
5: **while** not empty cand_streets **do**
6: calculate_crimes_thresholds(θ,Connected(k))
$C_{treshold} = \theta \times length(street)$
7: $M\_streets(k)$=Find_mean_streets whose $C_{real} \geq C_{treshold}$
8: $k = k + 1$
9: **end while**
10: return union M_street_1, M_street_2, ... M_street_k



(a) Sample dataset

Street Name	Street Length	Crime Number	Crime Thresholds	*Mean Streets*
A	5	10	10	
B	10	30	20	
C	10	0	20	Pruned
D	10	30	20	
E	10	30	20	
F	5	15	10	
G	10	15	20	Pruned
H	10	10	20	Pruned
J	10	155	20	
K	20	60	40	
L	20	30	40	Pruned
M	20	25	40	Pruned
N	20	25	40	Pruned
P	20	25	40	Pruned
Q	50	0	100	Pruned
R	50	0	100	Pruned
T	20	40	40	
U	50	0	100	Pruned
X	50	0	100	Pruned
Y	50	0	100	Pruned
Z	50	0	100	Pruned

(b) Singleton streets

Street Name	Street Length	Crime Number	Crime Thresholds	*Mean Streets*
AB	15	40	30	
AP	25	35	50	Pruned
BC	20	30	40	Pruned
BM	30	55	60	Pruned
DE	20	60	40	
DM	30	55	60	Pruned
DN	30	55	60	Pruned
DT	30	80	60	
EK	30	90	60	
EM	30	55	60	Pruned
ED	20	60	40	
EN	30	55	60	Pruned
FG	15	30	30	
KJ	30	205	60	
TG	30	55	60	Pruned

(c) Pair streets

Street Name	Street Length	Crime Number	Crime Thresholds	*Mean Streets*
ABC	25	40	50	Pruned
ABM	35	65	70	Pruned
DEK	40	120	80	
DTG	40	85	80	
EDT	40	100	80	
EKJ	40	245	80	
FGN	35	55	70	Pruned
FGH	25	40	50	Pruned

(d) Triple streets

Street Name	Street Length	Crime Number	Crime Thresholds	*Mean Streets*
DEKJ	50	275	100	
DTGN	60	110	120	Pruned
DTGH	50	95	100	Pruned
EDTG	50	115	100	
EDTGH	50	125	100	
EDTGN	70	140	140	Pruned
EDTGHJ	60	250	150	

(e) Quad, quintet, and sextet streets

Figure 6: Execution trace of the algorithm

3.3 An Execution Trace. The execution trace of the algorithm is given in Figure 6. The input dataset is a spatial graph which has 21 directed edges. Each edge has a name (street name), a length, and an aggregated crime value (Figure 6(a)). For example, the length of edge A is 5 units, the aggregated crime value is 20, and the edge is directed (e.g. one-way street). The total length of edges is 500 units and the total number of crimes on the network is 500.

The first step is to find the λ_{est} value (Definition 2.1). For the given dataset, $\lambda_{est} = 500/500 = 1$.

In the second step, the *mean street* density threshold $\theta(\lambda_{est}, \alpha)$ value is calculated for the given λ_{est} and confidence threshold $\alpha = 0.9$. For the given dataset, $\theta(\lambda_{est}, \alpha) = 2$.

In the third step, the idea is to find the crime thresholds for each street segment using the $\theta(\lambda_{est} = 1, \alpha = 0.9)$ value. The street segments having no fewer crimes than their crime thresholds are selected as *mean streets*. In Figure 6(b), the first column lists all the street segments, the second column gives the length of the segments and the third column gives the number of crimes of the street segments. For each street segment, the crime threshold is given in the fourth column. The crime threshold of an edge is the product of its length and the $\theta(\lambda_{est} = 1, \alpha = 0.9)$ value. For example the length of street segment A is 5 and its crime threshold is $5 \times \theta(\lambda_{est} = 1, \alpha = 0.9) = 5 \times 2 = 10$. The street segment A is a *mean street* since its actual number of crimes (which is 10) is equal to its crime threshold, 10.

When this process is applied to all the street segments, the segments that do not satisfy the criteria are pruned which are marked in Figure 6(b).

In the fourth step, the discovered singleton *mean streets* will be used to generate the pair candidate *mean streets* Figure 6(c). A singleton *mean street* will be extended by its neighboring street segments. For example, singleton mean street A has two neighbors, i.e. segments B and P, and the pair candidate list will include street sets of AB and AP. Similarly, singleton mean street B has four neighbors A, P, C, and M but only edges C and M will be used to generate pairs of B due to the directed property of mean street B. Figure 6(c) gives the complete list of candidate pair mean streets. After the pair street generations, their total lengths and number of crimes are calculated. If the total number of crimes of a street is less than its threshold, it is pruned. For example, in Figure 6(c), street sets AP, BC, BM, DM, DN, EM, and EN are pruned since $C_{real} < C_{threshold}$.

After the pair *mean streets* are discovered, they will be used to generate candidate triple streets. Next, the algorithm finds the neighbors of *mean streets*. Figure 6(d) gives the list of candidate triple streets and triple *mean streets*.

This process continues until there is no candidate pattern in the list. For the sample dataset given in Figure 6(a) the *mean streets* process finishes after the sextet streets are discovered (Figure 6(e)).

Finally, the algorithm outputs all size mean streets.

4 Analytical Evaluation.

4.1 Correctness and Completeness.

THEOREM 4.1. *The proposed Apriori-based and Graph-based algorithms are correct. In other words, if a street S is returned by the algorithms, then S is a mean street.*

Proof. The proof is easy to establish due to the pruning steps of *prune_unconnected_candidates*, and *find_mean_streets*, which weed out candidates not meeting the criteria

$$\text{number of crimes} \geq \text{length(candidate)} \times\theta(\lambda_{est}, \alpha)$$

THEOREM 4.2. *The proposed Apriori-based and Graph-based algorithms are complete, assuming that the street generator enumerates all candidate streets.*

Proof. The proposed algorithms are complete if they find all *mean streets* that satisfy the *mean street* density threshold θ which is found by confidence threshold α. We can show this by proving that none of the functions of the algorithms miss any patterns, i.e., filter out a *mean street*.

The *calculate* θ function does not miss any *mean streets*. It finds the value of θ using a given confidence threshold α and Poisson distribution for calculated λ_{est}

The *prune_unconnected_candidates* function does not miss any *mean streets*. It will check the connectivity of the pattern and unconnected combinations will be pruned. The input is *size* k candidate streets and the output is candidate connected *size* k patterns.

The *calculate_crime_thresholds* function does not miss any *mean streets*. This function calculates crime thresholds of streets using the parameter θ and the length of the street and does not prune any patterns.

The *find_mean_streets* function does not miss any *mean streets*. It prunes the street combinations whose number of crimes is less than the crime thresholds calculated in function *calculate_expected_crimes*.

The *generate_candidates* function does not miss any pattern and generates candidate *size* $k+1$ *mean streets*.

The *generate_all_possible_paths* function does not miss any pattern and generates all possible connected street segments and does not do any pruning.

The path generation algorithm enumerates all simple paths in a graph. This can be proved by induction on Step 7 of Algorithm 2. The algorithm at the p^{th} iteration appends every pair of existing paths of the form $(i, \ldots, p)$, $(p, \ldots, k)$ and the internal join node p is selected in order. So, the p^{th} iteration of the outermost loop finds all paths that has nodes $1, 2, \ldots, p$ as its internal nodes. Consider a simple path s in the graph G. If the node p is not an internal node of s, then this path is enumerated before the p^{th} iteration by inductive hypothesis, since new paths are generated by appending existing paths in the order of internal nodes. If p is an internal node of s, then it will be generated at the p^{th} iteration.

5 Experimental Evaluation.

In this section, we present our experimental evaluations of several design decisions and workload parameters on our proposed algorithms. We evaluated the behavior of both algorithms to answer the following questions:
- What is the effect of the user-defined threshold α?
- What is the effect of the network size?

Figure 7 shows the experimental setup to evaluate the impact of design decisions on the performance on both algorithms. Experiments were conducted on an Sun Solaris Workstation with 1.77 GHz CPU, 1GB RAM. The comparison metric is the execution time (cpu time). Real dataset experiments are conducted using C/C++ and synthetic dataset experiments are conducted using Matlab.

To evaluate the performance of the algorithms, real-wold and synthetic datasets were used.

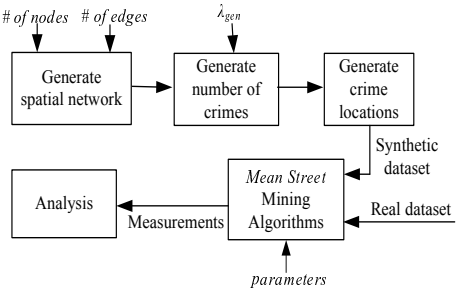


Figure 7: Experimental setup and design

1) Real Dataset:The real dataset contains crime incidences of a metropolitan city in United States for year 2006. This data has 1639 street segments or edges, 1348 intersections, and a total street length of 537km. Crimes are aggregated per street segment and the total number of crimes is 367.

2) Synthetic Dataset: The synthetic road network datasets were generated based on the network generator proposed by Cherkassky et. al. [5]. First, a spatial network was generated for V nodes and E edges where the length of each edge was assigned randomly. Five different synthetic datasets were generated for node values 10, 20, 30, 40, and 50. Unless otherwise stated, the ratio of number of edges to number of nodes was set to 3. Then, a Poisson pdf was used to generate N random crime numbers with network density parameter $\lambda_{gen} = 4$ for the spatial network. Finally locations of crimes were assigned randomly on the spatial graph.

5.1 Experiments with Real Dataset.

5.1.1 Effect of the Confidence Threshold. In the first experiment, we evaluated the effect of the confidence threshold on the execution times of both algorithms for the real dataset. The fixed parameters were number of nodes, number of edges, total number of crimes, and total length of road network and their values were 1348, 1639, 367, and 537km respectively. For the Apriori-based approach, a significant part of the computation time would be spent in generating candidate *mean streets* without looking at the connectivity of the edges. Experimental results show that the execution time variation in the Graph-based approach is less pronounced. This is more computationally efficient than the Apriori-based approach since only the connected paths are generated (Figure 8). The Apriori-based algorithm generates candidates without checking the graph connectivity, thus increasing the size of the search space. The execution time of the Graph-based approach decreases as the confidence threshold increases. It is also observed that the Apriori-based approach is computationally more expensive as the confidence threshold decreases because of the increase in the number of *mean streets* to be discovered.

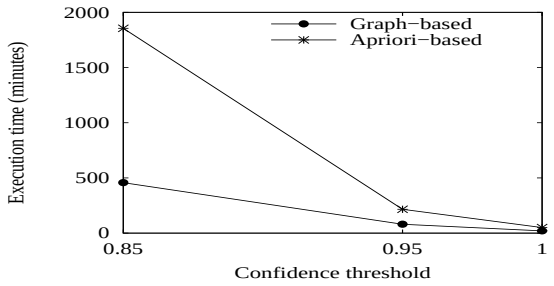


Figure 8: Effect of confidence threshold for real dataset

5.1.2 Effect of Network Size. In the second experiment, we evaluated the effect of the network size on the execution times of both algorithms for the real dataset. The confidence threshold was fixed at a value of 95%. We used three datasets and their specifications are given in Table 1. For the Apriori-based approach, a significant part of the computation time would be spent in generating candidate *mean streets* without looking at the connectivity of the edges. Experimental results show that the Graph-based approach is more computationally efficient than the Apriori-based approach since the search space is much larger in the case of the latter (Figure 9).

Table 1: Real datasets

	Dataset 1	Dataset 2	Dataset 3
# of edges	1639	1203	841
# of nodes	1348	902	629
# of crimes	367	272	171
total length (km)	537	399	332

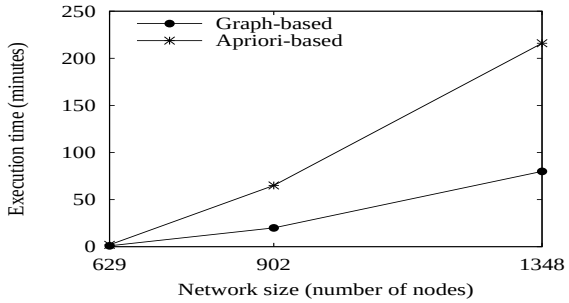


Figure 9: Effect of network size for real dataset

5.2 Experiments with Synthetic Dataset.

5.2.1 Effect of the Confidence Threshold. We evaluated the effect of the confidence threshold on the execution times of both algorithms for the synthetic dataset. The fixed parameters were number of nodes, number of edges, and number of crimes per unit length and their values were 50, 150, and 4 respectively. Experimental results show that the execution time variation in the Graph-based approach is less pronounced. This is because the Graph-based approach is more computationally efficient than the Apriori-based approach, since only connected paths are generated (Figure 10). The execution time of the Graph-based approach decreases as the confidence threshold increases. It is also observed that the Apriori-based approach is computationally more expensive as the confidence threshold decreases because of the increase in the number of *mean streets* to be discovered.

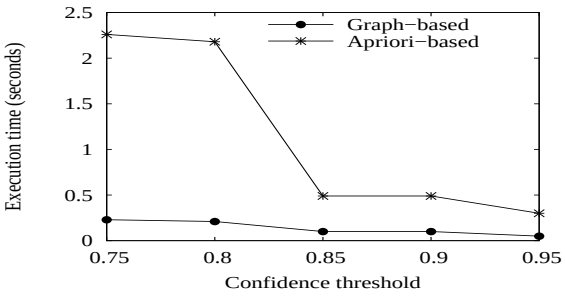


Figure 10: Effect of confidence threshold for synthetic dataset

5.2.2 Effect of Network Size. In this experiment, we evaluated the effect of the network size on the execution times of both algorithms for the synthetic dataset. The fixed parameter were the number of crimes per unit length and confidence threshold and their values were 4 and 90% respectively. Results show that the execution time in the Graph-based approach is more computationally efficient than the Apriori-based approach since the search space is much larger in the case of the latter (Figure 11).

6 Conclusions and Future Work.

We defined *mean streets* and the *mean street* discovery problem based on aggregated crime data on a spatial network, taking into account graph properties such as connectivity and directionality of the edges. The proposed method incorporates a statistical framework while discovering and quantifying *mean streets*. We also

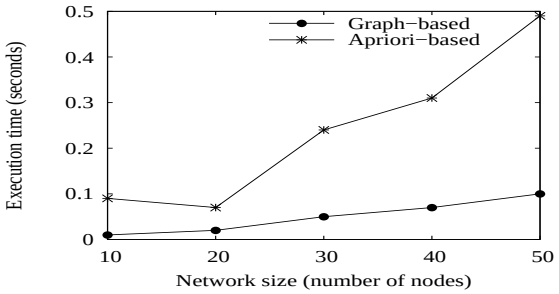


Figure 11: Effect of network size for synthetic dataset

proposed novel and computationally efficient methods for the discovery of *mean streets* and provided proofs for the correctness and the completeness of the proposed methods.

In our mixed process Poisson model we assumed that the point processes (crimes) would be independent of both location and time. This is often not the case. For future work, we will look at models which do not assume this independence and may be closer to real world patterns. We will explore new heuristics to improve the computational efficiency of the proposed algorithms to make them scalable to large road networks and the applicability of finding connected subgraphs in our proposed graph-based algorithm. Though the road network in the real dataset used in the evaluation happened to be sparse (average degree is less than 2) we will try to acquire denser graph datasets and study the performance of our algorithms. The proposed approach is dependent on edge (or segment length). In the future, we plan to explore different unit lengths or segment sizes, especially given the sensitivity of upper Poisson thresholds to unit lengths.

Alhough the proposed algorithms find the *mean streets* based on the aggregated values of crimes at various locations, they do not consider the possible temporal nature of the discovered patterns. For example, based on the time-variant availability of public transportation facilties (eg., buses, trains), the criminal route can vary over a day, resulting in a change in the *mean streets* over a time period. Considering the temporal aspects of mean streets might lead to an improvement in the efficiency and effectiveness of crime prevention measures such as patrolling. Discovery of time-dependent *mean streets* would require the underlying transportation network to be modeled as a spatio-temporal network. We will explore existing spatio-temporal network models such as the time aggregated graph, which models time variant attributes of networks by aggregating edge/node properties as time series on edges and nodes,

and formulate algorithms based on such models [9, 10].

7 Acknowledgments.

The authors would like to thank Kim Koffolt for her comments.

References

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

[2] P. Bickel and K. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Prentice Hall, 1977.

[3] P. R. Canter. Geographic information systems and crime analysis in baltimore county, maryland. In *Crime Mapping and Crime Prevention*, pages 157–192. Criminal Justice Press, 1997.

[4] C. Chekuri and S. Khanna. Edge disjoint paths revisited. *Proceedings of the ACM-SIAM*, 2003.

[5] B. Cherkassky, A. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Math. Prog.*, 73(2):129–174, 1996.

[6] J. E. Eck and et. al. Mapping crime:understanding hot spots. *US National Inst. of Justice (http://www.ncjrs.gov/pdffiles1/nij/209393.pdf)*, 2005.

[7] M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions, Third Edition*. Wiley, 2000.

[8] R. Floyd. Nondeterministic algorithms. *Journal of ACM*, 14:636–644, 1967.

[9] B. George, S. Kim, and S. Shekhar. Spatio-temporal Network Databases and Routing Algorithms: A Summary of Results . In *Proceedings of International Symposium on Spatial and Temporal Databases (SSTD'07)*, Boston, MA, USA, July 2007.

[10] B. George and S. Shekhar. Time Aggregated Graphs: A model for spatio-temporal network. In *Proceedings of the Workshops (CoMoGIS) at the 25th International Conference on Conceptual Modeling (ER2006)*, Tucson, AZ, USA, 2006.

[11] A. Getis and J. Ord. Local spatial statistics: An overview. In *Spatial Analysis: Modelling in a GIS Environment*, pages 261–277. GeoInformation International, Cambridge, England, 1996.

[12] O. Goldschmidt, D. Hochbaum, A. Levin, and E. Olonick. The sonet edge-partition problem. *Networks*, 41(1), 2003.

[13] J. Grandell. *Mixed Poisson Processes*. Chapman and Hall, 1997.

[14] J.Crawshaw and J. Chambers. *A Concise Course in Advanced Level Statistics*. Nelson Thornes, New York, 2001.

[15] T. Kamae. A systematic method of finding all directed circuits and enumerating all directed paths. *IEEE Trans. on Circuit Theory*, 14(2), June 1967.

[16] Y. Lee, H. Sherali, J. Han, and S. Kim. A branch-and-cut algorithm for solving an intra-ring synchronous optical network design problem. *Networks*, 35(3), 2000.

[17] N. Levine. *CrimeStat 3.0: A Spatial Statistics Program for the Analysis of Crime Incident Locations*. Ned Levine & Associatiates: Houston, TX / National Institute of Justice: Washington, DC, 2004.

[18] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *10th International Symposium on Spatial and Temporal Databases*, Boston, Ma, 2007.

[19] P. Mateti and N. Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal on Computing*, 5(1), 1976.

[20] J. Moller and R. D. Waagepetersen. *Statistical Inference and Simulation for Spatial Point Patterns*. Chapman and Hall, 2004.

[21] A. M. Mood, F. A. Graybill, and D. Boes. *Introduction to the Theory of Statistics, Third Edition*. McGraw Hill, 1974.

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

[23] A. Okabe, H. Yomono, and M. Kitamura. Statistical analysis of the distribution of points on a network. *Geographical Analysis*, 27, 1995.

[24] K. R. Parthasarathy. Enumeration of paths in digraphs. *Psychometrika*, 29(2), June 1964.

[25] J. H. Ratcliffe. The hotspot matrix: A framework for the spatio-temporal targeting of crime reduction. *Police Practice and Research*, 5(1):05–23, 2004.

[26] R. D. Reiss. *A Course on Point Patterns*. Springer, 1993.

[27] S. Roach. *The Theory of Random Clumping*. Methuen, London, 1968.

[28] F. Rubin. Enumerating all simple paths in a graph. *IEEE Trans. on Circuits and Systems*, 25(8), 1978.

[29] 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.

[30] A. Sutter, F. Vanderbeck, and L. Wolsey. Optimal placement of add/drop multiplexers. *Operations Research*, 46("5"), May 1998.

[31] J. Suurballe and R. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.

[32] R. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing*, 2(3), 1973.

[33] D. Torrieri. Algorithms for finding an optimal set of short disjoint paths in a communication network. *IEEE Trans. on Communications*, 40, 1992.