

Safe Navigation in Urban Environments

Esther Galbrun
Boston University
galbrun@cs.bu.edu

Konstantinos Pelechrinis
University of Pittsburgh
kpele@pitt.edu

Evimaria Terzi
Boston University
evimaria@cs.bu.edu

ABSTRACT

Advancements in mobile technology and computing have fostered the collection of a large number of civic datasets that capture the pulse of urban life. Furthermore, the open government and data initiative has led many local authorities to make these datasets publicly available, hoping to drive innovation that will further improve the quality of life for the city-dwellers. In this paper, we develop a novel application that utilizes crime data to provide safe urban navigation. Specifically, using crime data from Chicago and Philadelphia we develop a risk model for their street urban network which allows us to estimate the relative probability of a crime on any road segment. Given such model we define two variants of the SAFEPATHS problem where the goal is to find a short and low-risk path between a source and a destination locations. Since both the length and the risk of the path are equally important but cannot be combined into a single objective, we approach the urban-navigation problem as a biobjective shortest path problem. Our algorithms aim to output a small set of paths that provide tradeoffs between distance and safety. Our experiments demonstrate the efficacy of our algorithms and their practical applicability.

1. INTRODUCTION

A recent United Nations report states that more than 50% of the world's population currently lives in cities. This percentage is projected to increase to 70% by 2050 [8]. One of the main reasons for these levels of urbanization is the long-lived thought of cities as the paramount instrument for innovation and wealth creation. Nevertheless, there are side effects attached to this phenomenon. Cities have become the main source of crimes, diseases and pollution, significantly deteriorating the quality of life of their dwellers.

During the last years, local authorities have collected a large number of civic datasets that capture various aspects of urban life. These datasets include information for street constructions, crimes, bicycle routes, etc. The open government and data initiative from President Obama's administration [7] has further led many local authorities to systematically collect, organize and publicize such datasets. Besides the transparency of federal and local government operations, this initiative seeks the development of innovative ser-

vices that will impact people's lives, from the people themselves.

In this paper, we take steps towards this direction. In particular, we focus on one of the major aforementioned problems present in almost every megacity today: crimes and public safety. We develop a novel application aiming to identify *safe* and *short* paths for people to take when they navigate the city. For this purpose, we take into consideration both the spatial constraints imposed by the structure of the city's road network as well as the criminal activity in the city. Ideally, prioritizing the safety of dwellers, one would like to provide the user with the safest path from origin s to destination t . However, such a path may incur a significant penalty in the total distance that needs to be traveled. Conversely, the shortest path between s and t may be risky. Therefore, we approach the problem of safe urban navigation as a bicriteria optimization problem where the goal is to provide users with a small collection of paths, each offering a different tradeoff between distance and safety. Possibly, none of the available walking options might be satisfying to the user, who might consider that the safest path is still too risky, for instance, and in light of this information decide to use alternative means of transportation (e.g., taxi).

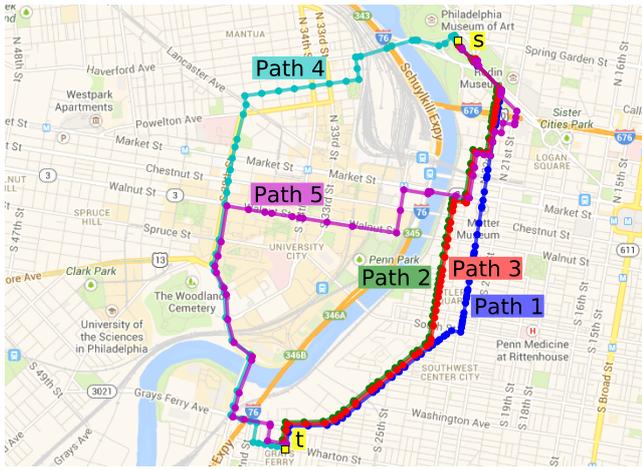
As an example, consider the scenario illustrated by Figure 1: a city dweller is at the Philadelphia Museum of Art (s) and wants to return to her home on Wharton Street (t). Her shortest way to home is given by Path 1. However, according to the crime model we develop in Section 2, the itinerary indicated as Path 5 constitutes her safest option, which is also about 1.5 times longer than Path 1. Instead, our art lover may prefer one of the intermediate routes (Paths 2–4), that offer various tradeoffs between distance and risk.

Our approach in a nutshell: Our approach includes a modeling and an algorithmic component. As part of the former, we develop a framework for the crime risk on street urban networks. We focus on the US cities of Chicago and Philadelphia. After exporting their street networks from OpenStreetMap (OSM) into a graph format we use publicly available crime datasets to assign a risk score to each edge (i.e. street segment).

As part of the algorithmic component, we define the SAFEPATHS problem as a biobjective shortest path problem where the goal is to output a *collection of paths* that offer different tradeoffs between their length and the associated risk. Of course the space of all possible paths can be extremely large and showing all such paths to the users may be overwhelming. Thus, our goal is to select a *small* subset of paths that gives a good summary of the solution space and the available tradeoffs. We study two variants of this generic problem that differ on their definition of the risk of a path. In the first variant, the risk of the path is defined as the total probability of a crime happening along all its segments. In the second variant, the risk of the path is defined as the maximum risk to which the user is exposed on any segment. The key challenge in designing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright is held by the author/owner(s).
UrbComp'14, August 24, 2014, New York, NY, USA.



	Path 1	Path 2	Path 3	Path 4	Path 5
Length ℓ (m)	3955	4027	4060	4922	5988
Risk r ($\times 10^{-3}$)	2.32	2.02	2.01	1.71	1.70

Figure 1: An illustrative example of safe urban navigation. The routes depicted as Paths 1–5 offer various compromises between distance and risk for traveling between s and t .

algorithms for these problems stems from the fact that they aim to explore and summarize a possibly exponential solution space. The algorithms we present can report a representative set of paths in a matter of seconds. Moreover, we enhance these algorithms with early-stopping criteria which, in practice, lead to 4-fold reduction of the running time with minimal sacrifices in the information content they provide to the user.

Roadmap: The rest of the paper is organized as follows. Section 2 describes the civic datasets as well as the risk model we developed for the urban road networks. Then, in Section 3, we provide the necessary notation and formally define the SAFE PATHS problem using the developed risk model, before describing our algorithms to solve two specific instances of the problem in Section 4. In Section 5 we present our experimental results, and after reviewing the related work in Section 6, we conclude in Section 7.

2. MODELING URBAN SAFETY

Throughout the paper, we will represent the road network using an undirected graph $G = (V, E)$. The nodes of the graph, V , represent intersections and the edges, E , represent the road segments that connect intersections. The focus on undirected graphs is due to the fact that our framework is targeted primarily to pedestrians, since they are more exposed to risks in urban areas. Clearly, such users can traverse street segments in both directions. However, we emphasize that this assumption is not restrictive and the proposed framework can also be used with directed graphs.

Each road segment $e \in E$ is associated with two (unrelated) types of weights: its *length*, denoted by $\ell(e)$, and its *risk*, denoted by $r(e)$. The length of edge e corresponds to the actual distance between the two intersections connected by the associated street segment, while the risk of edge e should be interpreted as the *probability* that a crime will be committed on that segment.

We now describe how this model can be obtained. More specifically, we explain how we combine information from different civic datasets in order to build a risk model for two US cities.

2.1 Urban road networks

We extract the actual road network of a city using OpenStreetMap (OSM) [3]. OSM is a crowdsourcing-based open platform, which aims to make the world map freely available. The OSM community has developed a number of tools that facilitate the integration of OSM maps with a variety of applications. For our purposes we make use of the `osm4routing` parser [5], that exports the map of interest from an OSM format (appropriate for traditional GIS systems) to a graph-based format (appropriate for graph-based algorithms).

In particular, each node v in the exported graph G corresponds to an intersection, while each edge e represents a street segment that connects two intersections. `osm4routing` provides additional information about the intersections and the street segments as node and edge attributes. Specifically, the geographical location (latitude/longitude pair) of each node is provided. Each edge e is annotated with a variety of features. The following ones are of particular interest for our study:

- *Length* $\ell(e)$, the physical length of the corresponding street segment. It is the length value we associate with every edge in our input graph.
- *Geometry* $\Gamma(e)$, a sample set of discrete latitude/longitude points from the street segment. It essentially provides information about the actual geographic *shape* of the street segment.
- *Accessibility flags*, indicators that capture the accessibility of a street segment by car, bicycle and foot.

Here, we consider the pedestrian street network of Chicago and Philadelphia. We obtained the OSM maps for the metropolitan areas of these two cities from Metro Extracts [4]. The resulting two graphs, Chicago and Philadelphia, representing the street networks of the metropolitan areas of these two cities, respectively, have fairly similar sizes: Chicago comprises 57 998 nodes and 91 695 edges, compared to 55 234 nodes and 82 676 edges in Philadelphia. In both cities, most edges span a short geographic distance, that is, less than 250m. There are few edges spanning long distances, 4km and above, that most likely correspond to highways.

2.2 Crime and safety

A risk model for the urban road network is essentially an assignment of a risk score r to each edge that is proportional to the probability of a crime happening on the corresponding street segment. Note that we do not intend to infer the exact risk probability of individual edges with this model. Rather, we are interested in the relative values of r for different edges and the fact that they can be interpreted as probabilities.

In addition to the above, we also use two civic datasets made available by the cities of Chicago [1] and Philadelphia [2] that encode information about crimes in the two cities for the one year period between October 2012 and September 2013. Their formats do not align, as it is true with the majority of civic datasets made available from different cities even when they aim to convey the same information. Nevertheless, some basic information that both of our datasets include are encoded in the tuple:

`< Date, Time, Latitude, Longitude, TypeofCrime >`.

Criminal activity: To build our model we will first use the geographic coordinates of the crime incidents to compute a *spatial density* for the crime activity by applying a Gaussian kernel density estimation (KDE). Given n points of crime incidents c_1, c_2, \dots, c_n on a 2-dimensional plane, the Gaussian kernel estimates the density

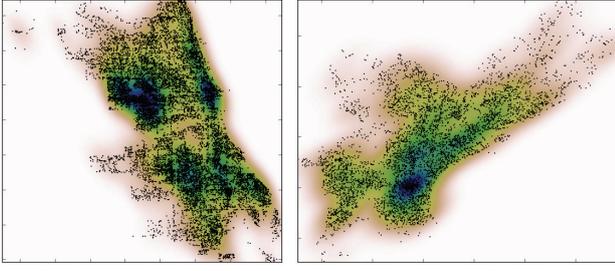


Figure 2: KDE for crime activity in Chicago (left) and Philadelphia (right).

of criminal activity density at point \mathbf{p} as:

$$\lambda(\mathbf{p}) = \frac{1}{nh^2} \sum_{i=1}^n \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{p}\|_2^2}{2h^2}\right), \quad (1)$$

where $\|\mathbf{c}_i - \mathbf{p}\|_2$ is the Euclidean distance between points \mathbf{c}_i and \mathbf{p} and h is the bandwidth used. The bandwidth h dictates the spread of the Gaussian kernel that is centered at each datapoint and hence, it controls the smoothness of the estimated density. Small values of h can capture more detail while large values of h lead to smoother estimation. In other words, it is the analogue of the bin width of a histogram. We determine h according to Scott's rule [32]. Figure 2 visualizes the estimated crime activities for the two cities. As we can observe, crimes are not uniformly distributed over the metropolitan areas of the two cities but there are *hotspots* of crimes, something expected based both on our experience as city-dwellers as well as on existing literature in a variety of fields (e.g. [13, 31]).

Once the density function is estimated, we evaluate it on the actual road segments. For this purpose, we make use of the geometry $\Gamma(e)$ of each edge e of the road network. Evaluating Equation (1) on every point $\xi_i \in \Gamma(e)$ and summing up we obtain the crime activity density $\Lambda(e)$ on road segment e : $\Lambda(e) = \sum_{\xi_i \in \Gamma(e)} \lambda(\xi_i)$. $\Lambda(e)$ is proportional to the probability of *observing* a crime incident on edge e .

Risk model: The crime activity is typically higher in denser, popular areas simply because there are more opportunities for criminal activities [22]. Hence, from a risk perspective we should *normalize* the crime activity with the population/mobility density in the corresponding urban environment. Simply put, for a more accurate definition of the risk imposed to a city-dweller while traversing street segment e , one would also need to estimate the spatial *urban activity density* on e . The latter essentially captures the population density observed on e and can be computed using fine-grained mobility traces (e.g., from call detail records, GPS-traces etc.). In particular, given m points $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m$ on the 2-dimensional plane that capture the locations where dwellers have been observed, we can estimate the activity density $\alpha(\mathbf{p})$ at point \mathbf{p} using Equation (1). Similarly to above, we can further estimate the urban activity density on edge e as: $A(e) = \sum_{\xi_i \in \Gamma(e)} \alpha(\xi_i)$. We can then use the logistic function of the ratio between the crime and urban activity densities to define the risk $r(e)$ associated with edge e as:

$$r(e) = \frac{1}{1 + e^{-\Lambda(e)/A(e)}} \quad (2)$$

For our current study, we do not have access to detailed urban mobility traces and hence, in our experiments we will directly make use of the spatial crime density. In particular, we compute the risk

weights for every edge e as the normalized densities:

$$r(e) = \frac{\Lambda(e)}{\sum_{e' \in G} \Lambda(e')}. \quad (3)$$

Temporal dimension: Finally, criminal activity has a strong temporal component as well. The temporal dimension can be incorporated in our model by filtering the data points that we use for training the KDE, in particular, by focusing the crime incidents (and the mobility traces) that have been recorded within a time window Δt prior and after the query time during the previous δ days, based on the time t of the path query from a user. Nevertheless, identifying the optimal time granularity (i.e., Δt and δ) is beyond the scope of our work. While incorporating the time dimension is straightforward, it requires fine tuning of parameters that can be highly dependent of the final application scenario. Therefore, we ignore the temporal dimension in our experiments.

3. PROBLEM DEFINITION

We now turn to an application exploiting the model presented above to provide safe pedestrian navigation in urban environments. In this section, we present the necessary notation and then formally define the SAFEPATHS problem and its variants.

Preliminaries: To reiterate, we consider a graph $G = (V, E)$ with two (unrelated) types of weights on its edges, the *length* $\ell(e)$, and the *risk* $r(e)$.

A path P on G is defined as a connected sequence of edges; a path P from a particular source, s , to a particular destination t is a connected sequence of edges such that the first edge in the sequence has its starting point in s and the last edge in the sequence has its ending point in t . When we want to specify the source and destination nodes of a path P , we denote it by $P_{s,t}$.

Clearly, the *length* of a path P is the sum of the lengths of its edges; that is, $\ell(P) = \sum_{e \in P} \ell(e)$. The *risk* of a path P , $r(P)$, is also a function of the risks of its edges, but we consider two alternative definitions for it: the *total-risk*, denoted by $r_t(P)$ and the *max-risk*, denoted by $r_m(P)$.

On one hand, $r_t(P)$ is the probability of a crime happening anywhere along the path, or in other words, on any edge of P . More specifically, it is defined as follows:

$$r_t(P) = 1 - \prod_{e \in P} (1 - r(e)). \quad (4)$$

On the other hand, $r_m(P)$ is the highest crime probability encountered on any single edge along the path P , that is,

$$r_m(P) = \max_{e \in P} r(e). \quad (5)$$

Note that both $r_t(P)$ and $r_m(P)$ have a probability interpretation. However, while $r_m(P)$ depends only on the most risky edge of the path, $r_t(P)$ involves the risk of all the edges in P . In both cases, we look for safe paths. That is, we want to minimize $r(P)$, whether it is instantiated as $r_t(P)$ or $r_m(P)$.

The SAFEPATHS problem: At a high level, the SAFEPATHS problem takes as input the road network $G = (V, E)$ together with a pair of source-destination nodes, (s, t) , and its goal is to provide to the user a (set of) short *and* safe paths between s and t .

The difficulty with this high-level problem definition is that clearly, the shortest path (in terms of length) is not necessarily the one with the smallest risk and vice versa. Therefore, the problem we actually want to solve is a bicriteria optimization problem, where the goal is to minimize both the length as well as the risk of the reported

path. Of course, there are multiple ways of formalizing, and subsequently, solving such a bicriteria optimization problem. One way to approach it is by combining the two objectives, the length and the risk of the path, into a single objective and ask for the path $P_{s,t}$ minimizing the weighted combination:

$$P_{s,t} = \arg \min_{P'_{s,t}} (\alpha \times \ell(P'_{s,t}) + \beta \times r(P'_{s,t})),$$

where α and β are appropriately-tuned real-valued coefficients and $r(P)$ encodes the max or the total risk of P .

Although such a formulation could be an option, it has its shortcomings. First, it is not straightforward to appropriately define the parameters α and β since they depend on the user preference. In this particular case, setting them is even more difficult because $\ell(P)$ and $r(P)$ are not measured in comparable units. This latter point also highlights that finding a single path that optimizes the above objective does not have a direct and intuitive interpretation since it is a single solution that optimizes a summation of distances and probabilities.

Instead of combining the two objectives into a single one and finding a unique solution with respect to that composite objective, we take a different approach. We look for a small set of paths that provide tradeoffs between the two objectives.

Every path P from s to t is a 2-dimensional point with its x -coordinate corresponding to $\ell(P)$ and its y -coordinate corresponding to $r(P)$ (see Figure 3 for an example). Clearly, there are many paths from s to t , corresponding to different 2-dimensional points. Therefore, in order to exhaustively explore the solution space one needs to consider all possible paths and investigate the tradeoffs they offer in terms of length versus risk. From a practical perspective, such an exhaustive exploration of the solution space is infeasible, simply because the number of possible paths can be exponential in the input size.

However, if \mathcal{P} is the set of all possible paths from s to t , then not all of these paths are of interest to the user. For instance, consider two paths P_1 and P_2 , such that $\ell(P_1) \leq \ell(P_2)$ and $r(P_1) < r(P_2)$: P_1 is better than P_2 both in terms of length and risk. In this case, we say that P_2 is *dominated* by P_1 . Obviously, solutions in \mathcal{P} that are dominated by other solutions need not be considered. Since the notion of *domination* of paths is important for the rest of the discussion we provide a formal definition below.

DEFINITION 1. *Given two paths P and P' and assuming that their lengths and risks can be computed using functions $\ell(\cdot)$ and $r(\cdot)$, respectively, which are to be minimized, we say that P dominates P' if $\ell(P) \leq \ell(P')$ and $r(P) \leq r(P')$ and at least one of the two inequalities is strict, that is, either $\ell(P) < \ell(P')$ or $r(P) < r(P')$.*

We are also not interested in returning *equivalent* paths, i.e. paths P and P' for which $\ell(P) = \ell(P')$ and $r(P) = r(P')$. In such cases we break ties arbitrarily.

Given the above, we are now ready to define the SAFEPATHS problem:

PROBLEM 1 (SAFEPATHS). *If \mathcal{P} is the set of all possible paths from s to t , our goal is to select a small subset of these paths $\mathcal{S} \subseteq \mathcal{P}$ such that for every path $P \in \mathcal{S}$, P is neither dominated by nor equivalent to any other path $P' \in \mathcal{S}$.*

Observe that in the above discussion we kept the definition of the risk of the path generic, using $r(P)$ to denote indistinctly the max risk as well as the total risk. We call the instance of the SAFEPATHS problem where the risk of the path is computed using $r_i(P)$ (Equation (4)) the TOTAL-PATHS problem. On the other hand, when the

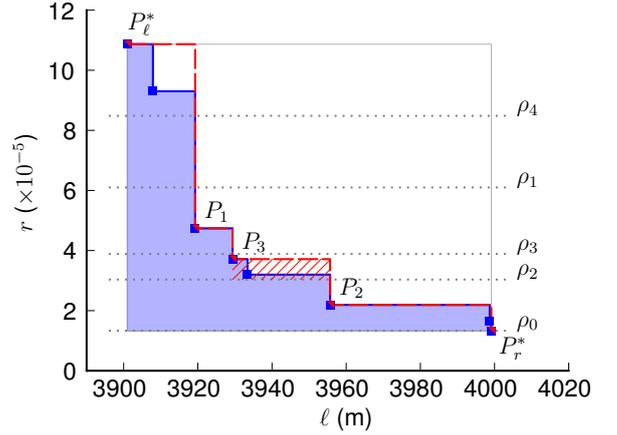


Figure 3: The 2-dimensional representation of paths from the solution space \mathcal{P} of a SAFEPATHS instance.

risk of P is computed using $r_m(P)$ (Equation (5)), we call the corresponding problem the MAX-PATHS problem.

Observe that Problem 1 asks for a “small” subset of nondominated paths. One can naturally wonder what the right number of paths is. Here we aim to provide the end user with a set of paths that she can easily parse. At the same time, this set should adequately represent the space of nondominated paths.

4. ALGORITHMS

In this section, we present algorithms for TOTAL-PATHS and MAX-PATHS. Given a source-destination pair (s, t) , the common goal of these algorithms for both problems is to efficiently explore the space of nondominated paths from s to t .

These algorithms can be more easily understood if we visualize every path $P \in \mathcal{P}$ as a 2-dimensional point, with its x -coordinate corresponding to $\ell(P)$ and its y -coordinate corresponding to $r(P)$. An example of a 2-dimensional representation of some of the paths of \mathcal{P} is shown in Figure 3.

For a fixed (s, t) pair, there are two important paths: the shortest path, denoted by P_ℓ^* , and the safest path, denoted by P_r^* . If the two coincide, then the point that corresponds to the shortest and safest solution dominates all others, and the algorithms report this as the only solution.

When these two paths are distinct, they are both nondominated. Furthermore, they define the “region” of the solutions in their 2-dimensional representation. In fact, all the nondominated solutions will have lengths in the interval $[\ell(P_\ell^*), \ell(P_r^*)]$ and risks in the interval $[r(P_r^*), r(P_\ell^*)]$.

We propose recursive algorithms for both versions of the SAFEPATHS problem that share the same high-level principle. Starting with the shortest and the safest paths, P_ℓ^* and P_r^* , they consider in each iteration a pair of nondominated paths P_u and P_l and search for an intermediate nondominated path P_i inside the subintervals $[\ell(P_l), \ell(P_u)]$ and $[r(P_u), r(P_l)]$ that they define within the original intervals. This process is repeated recursively with the pairs (P_u, P_i) and (P_i, P_l) until convergence.

Despite their common high-level description, the details of our algorithms for TOTAL-PATHS and MAX-PATHS differ. We describe them in turn in Sections 4.1 and 4.2. We also present speedups which can be applied to both algorithms and lead to significant computational savings, in Section 4.3

<pre> Sum-Recursive($G, \ell(), r()$) 1: $P_\ell^* \leftarrow \text{Dijkstra}(G, \ell())$ 2: $P_r^* \leftarrow \text{Dijkstra}(G, r())$ 3: $\mathcal{S} \leftarrow \{P_\ell^*, P_r^*\}$ 4: $\text{sum-rec}(P_\ell^*, P_r^*, \mathcal{S})$ 5: return \mathcal{S} </pre>	<pre> Routine $\text{sum-rec}(P_u, P_l, \mathcal{S})$ 1: $\lambda \leftarrow (r(P_u) - r(P_l)) / (\ell(P_u) - \ell(P_l))$ 2: $\forall e \in E : f(e) = r(e) - \lambda \ell(e)$ 3: $P_i \leftarrow \text{Dijkstra}(G, f())$ 4: if $P_i \neq P_u$ and $P_i \neq P_l$ then 5: $\mathcal{S} \leftarrow \mathcal{S} \cup \{P_i\}$ 6: $\text{sum-rec}(P_u, P_i, \mathcal{S})$ 7: $\text{sum-rec}(P_i, P_l, \mathcal{S})$ </pre>	<pre> Routine $\text{max-rec}(P_u, P_l, \mathcal{S})$ 1: $\rho \leftarrow (r(P_u) + t(P_l)) / 2$ 2: if $\exists e \in E, t(P_l) < r(e) < \rho$ then 3: $P_i \leftarrow \text{Dijkstra}(G_{r(e) < \rho}, \ell())$ 4: if $P_i \neq P_l$ then 5: $\mathcal{S} \leftarrow \mathcal{S} \cup \{P_i\}$ 6: $\text{max-rec}(P_i, P_l, \mathcal{S})$ 7: $\text{max-rec}(P_u, P_i, \mathcal{S})$ </pre>
---	--	---

Figure 4: The Sum-Recursive algorithm (left) associated to the `sum-rec` routine (middle) for the TOTAL-PATHS problem and the `sum-rec` routine (right) for the MAX-PATHS problem.

Computing shortest paths: In all our algorithms we will often use the Dijkstra shortest path routine. $\text{Dijkstra}(G, f())$ will indicate that we run the Dijkstra shortest-path algorithm on input graph G with weights on edges defined by function $f()$. That is, the output of $\text{Dijkstra}(G, f())$ is a path P in G minimizing $\sum_{e \in P} f(e)$.

4.1 Algorithms for the TOTAL-PATHS

First, let us take a closer look at the two objectives of the TOTAL-PATHS problem. For every path P the length of the path, $\ell(P)$, is written as a summation of the lengths of the edges in P . Moreover, the total risk of the path, $r_t(P)$, given in Equation (4), can equivalently be minimized as a summation, namely, as the sum of the negative logarithms of one minus the risk of the edges along the path. Therefore, both our objectives can be computed as summations of (appropriately-defined) weights of the graph edges.

The solution space: Exploring the solution space of such “sum-sum” biobjective shortest path problem is a long-studied research topic and there exist many approaches that aim to compute the set of nondominated paths [24] or solve variants of this problem [28, 17]. However, there can be exponentially many nondominated paths. In addition to being expensive to compute, reporting such a solution set to the end user can be very overwhelming in practice and thus, unrealistic and impractical.

Hence, we consider the following alternative: if \mathcal{P} is the set of all paths between s and t , we are content with reporting the *lower convex hull* of \mathcal{P} , defined with respect to the 2-dimensional representation of the points. We denote as \mathcal{S} the set of points in the lower convex hull, also known as the *linear path skyline* [35]. Clearly, $\mathcal{S} \subseteq \mathcal{P}$ and all points in \mathcal{S} correspond to nondominated paths.

The Sum-Recursive algorithm: The pseudocode of this algorithm for finding \mathcal{S} is given in Figure 4. At a high level, the algorithm is a recursive algorithm that repeatedly calls the `sum-rec` routine. This routine takes as input two nondominated paths P_u and P_l and assigns to every edge e of the original graph a composite weight $f(e) = r(e) - \lambda \ell(e)$, where λ is the gradient of the line between paths P_u and P_l . The shortest path P_i of graph G with weights assigned by $f()$ is computed by $\text{Dijkstra}(G, f())$. This new path is added to the set \mathcal{S} and is used for recursively calling the `sum-rec` routine using as input the pair of paths (P_l, P_i) and the pair (P_i, P_u) . Paths P_ℓ^* and P_r^* are used as the initial pair, and they are also members of \mathcal{S} .

By the results of Mehlhorn and Ziegelmann [28], the set of paths \mathcal{S} reported by the Sum-Recursive algorithm upon termination is the lower convex hull of the solution space \mathcal{P} . If the algorithm reports $|\mathcal{S}|$ points, it further requires $O(|\mathcal{S}|)$ calls to the `Dijkstra` routine. Thus, the worst-case running time of Sum-Recursive is $O(|\mathcal{S}|(|E| + |V| \log |V|))$.

4.2 Algorithms for the MAX-PATHS

Again, let us take a closer look at the two objectives of the MAX-

PATHS problem. For every path P the length of the path $\ell(P)$ is again the summation of the lengths of the edges in P . However, now the max-risk of the path, $r_m(P)$, is the *maximum* of the risks of the edges in P (cf. Equation (5)). Therefore, one of our objectives can be written as a summation of edge lengths and the other as the max of the edges risks, resulting in a “sum-max” biobjective problem.

The solution space: This “sum-max” biobjective problem turns out to have a more structured solution space than the “sum-sum” problem we studied in the previous section. More specifically, any set of nondominated and nonequivalent paths in the solution space of the MAX-PATHS has size polynomial to the size of the input. Indeed, since the risk of a path corresponds to the risk of a single edge along the path, there can be only as many distinct values for the path risks as there are edges in the network and there can be at most one nondominated path for every such distinct value.

The Max-Exhaustive algorithm: The exhaustive algorithm for finding the nondominated paths is straightforward: first, find P_ℓ^* and report this as a nondominated path. Then, identify in P_ℓ^* the edge with the maximum risk and remove from the original graph all edges with risk greater or equal to that value. Find again the shortest path in this reduced subgraph and add the new path into the set of nondominated solutions. Iterate the above process until the source and target are no longer connected. This algorithm, that we call `Max-Exhaustive`, has at most $O(|E|)$ repetitions and each repetition requires running `Dijkstra` on the remaining graph, resulting in an $O(|E|^2 + |E||V| \log |V|)$ running time.

The Max-Recursive algorithm: Instead of going over possible risk thresholds in decreasing order, like `Max-Exhaustive`, `Max-Recursive` proceeds by recursion, similarly to `Sum-Recursive`, but replacing the `sum-rec` routine by the `max-rec` routine shown in Figure 4 (right).

At line 3, `Max-Recursive` runs the `Dijkstra` routine on $G_{r(e) < \rho}$, the subgraph obtained from G after removing all edges with risk greater or equal to ρ . By doing so, it identifies the shortest path P_i in G such that $r_m(P) < \rho$. For algorithmic purposes, we need to remember the threshold with which a path P has been discovered, which we denote by $t(P)$. Indeed, upon finding P_i , we know that there is no nondominated path with risk between $t(P_l)$ and $r(P_i)$, so we can leave out this range. Given two paths P_u and P_l , `max-rec` considers as risk threshold ρ the mean of $r(P_u)$ and $t(P_l)$. If there exist edges with risk between $t(P_l)$ and ρ , and hence potentially paths with r_m inside that range, the `Dijkstra` routine is run to identify the shortest such path. `max-rec` iterates with P_u and the new path, as well as with the new path and P_l , if they differ.

Another difference between the two algorithms, `Sum-Recursive` and `Max-Recursive`, is the computation of the *safest* path. While in the sum variant P_r^* was obtained by simply running $\text{Dijkstra}(G, r())$, here it is computed as follows. First, we compute the minimum spanning tree (MST) of the graph considering the risks of the edges

as the weights. The risk $\rho_{s,t}^*$ of the safest path between a given pair (s, t) is the maximum risk encountered along the unique path from s to t in this spanning tree. Then, we compute the shortest path verifying this risk threshold by running the Dijkstra routine on $G_{r(e) \leq \rho_{s,t}^*}$ to obtain P_r^* . Note that its length will generally differ from the length of the path connecting s and t in the spanning tree.

Figure 3 illustrates a run of this algorithm. First, the shortest path (P_ℓ^*) and safest path (P_r^*) are found, then ρ_1 is used as threshold, finding P_1 . The next thresholds are ρ_4 and ρ_2 in the upper and lower side respectively, and so on.

Since we are not looking for a unique solution we explore both sides after a binary split. Thus, the worst-case complexity of this approach is the same as the exhaustive search. In fact, the advantage of `Max-Recursive` only appears when combined with the approximation speedup discussed in the next section, which cannot be applied to `Max-Exhaustive`.

4.3 Computational speedups

In this section we describe our strategies for speeding up the above algorithms, applicable to both versions of the `SAFEPATHS` problem.

Approximation via early stopping: The running time of our algorithms for both versions of the `SAFEPATHS` problem depend linearly on the number of points in the set of paths returned by the algorithms. Therefore, one way of speeding up the algorithms is by reporting a subset of those paths. Hence, by restricting the number of recursions in `Sum-Recursive` and `Max-Recursive` we contribute to both of our aforementioned objectives, i.e., reducing the runtimes and limiting the size of the reported solution sets.

Specifically, consider the shortest and the safest paths with 2-dimensional representations $(\ell(P_\ell^*), r(P_\ell^*))$ and $(\ell(P_r^*), r(P_r^*))$ respectively, defining a rectangle of area A . Before calling the recursive routine with two paths P_u and P_l , we check the area of the unexplored rectangle between them. If this area becomes smaller than a chosen fraction γ of A we stop the iteration. With `sum-rec`, the unexplored rectangle for paths P_u and P_l is delimited by $(\ell(P_u), r(P_u))$ and $(\ell(P_l), r(P_l))$. With `max-rec`, it is delimited by $(\ell(P_u), r(P_u))$ and $(\ell(P_l), t(P_l))$, since we know there is no nondominated path with risk between $t(P_l)$ and $r(P_l)$. For instance, consider the scenario in Figure 3 with $\gamma = 0.1$. Since the hatched area is smaller than one-tenth of the total area A of the solution space (delimited by a solid grey line), the recursive routine is not called for P_3 and P_2 . In summary, the choice of γ directly limits the number of iterations and the number of paths in the solution; a larger γ results in fewer paths returned and shorter runtimes.

Pruning: Our pruning strategy, which we call the `Ellipse` pruning, is based on the following observation: any path between s and t that is longer than the safest path, will be dominated by it.

This observation has the following consequence: if L is the length of the safest path between s and t , then any node u such that the sum of the distances “as the crow flies” from s to u and u to t is greater than L cannot be part of the solution. This criterion defines an *ellipse* with foci s and t and major axis of length L . Hence, after finding P_r^* we can safely remove any node that lies outside the ellipse, thereby reducing the size of the graph processed in subsequent calls to the Dijkstra routine.

We combine the `Ellipse` pruning with another simple pruning technique: we split the area that corresponds to the city (or neighborhood) of interest into a grid and project the graph G on this grid. Then, instead of considering the original graph we only consider the grid cells that are included into or overlap with the ellipse.

Note that the `Ellipse` pruning reduces the size of the input graph, without sacrificing the accuracy. That is, the set of nondom-

inated paths before and after the `Ellipse` pruning remains the same.

5. EXPERIMENTS

In this section, we present an experimental assessment of our proposed methods. Our main objective is to evaluate the practical utility of our algorithms. We focus on two aspects: (i) the *representativeness* of the small set of nondominated paths that our algorithms return and (ii) their runtime.

We build our evaluation benchmark by sampling node pairs from both cities. To ensure a good coverage of scenarios involving short distances, which are of particular interest for a pedestrian urban navigation system, we perform distance-aware sampling. More specifically, we sample 100 nodes for each city. Then, for each of these source nodes, we divide the remaining nodes into classes based on their distance “as the crow flies” from the source. Concentric circles around the source that divide the plane into six distance classes further denoted by D_0, D_1, \dots, D_5 . With this setting we sample 4 targets from each class, obtaining a total of 2400 (s, t) pairs for each city.

We run our algorithms for the `TOTAL-PATHS` and the `MAX-PATHS` problems for each of the pairs on their respective road network. For each run, we obtain a set of paths offering different tradeoffs between distance and safety. For instance, Figure 1 depicts an actual solution of 5 paths to the `TOTAL-PATHS` problem for a pair of nodes in Philadelphia.

The 2-dimensional representation of a solution consisting of 8 paths to `MAX-PATHS` for a pair of nodes in Chicago is plotted in blue in Figure 3. As we explained previously, a path P can be represented in a 2-dimensional space by a point whose x and y -coordinates correspond to $\ell(P)$ and $r(P)$ respectively. Given a pair of nodes, the shortest and safest paths joining them, P_ℓ^* and P_r^* , define a rectangle R . A solution to the `SAFEPATHS` problem is a set of paths \mathcal{S} including P_ℓ^* and P_r^* . In the 2-dimensional space, \mathcal{S} can be represented with a curve that starts from P_ℓ^* , ends at P_r^* and passes through intermediate nondominated paths in \mathcal{S} across R . Formally, given a solution $\mathcal{S} = (P_\ell^*, P_1, P_2, \dots, P_r^*)$, where the paths are ordered by increasing length, the corresponding curve is defined by: $(\ell(P_\ell^*), r(P_\ell^*))$, $(\ell(P_1), r(P_1))$, $(\ell(P_2), r(P_2))$, \dots , $(\ell(P_r^*), r(P_r^*))$.

Based on this representation, each solution is associated with an area: the area of R located under the curve. Continuing on our example in Figure 3, the area associated with the 8-paths solution is colored in blue. Paths offering good tradeoffs correspond to points close to the bottom left corner of R . Hence, solutions containing good tradeoffs are associated with small areas (relative to the total area of R).

Approximations: The representativeness of our algorithms can be captured through their goodness of approximation as compared to the exact algorithms for the corresponding problems. In particular, we compare the exact algorithms `Sum-Recursive` (for `TOTAL-PATHS`) and `Max-Exhaustive` (for `MAX-PATHS`), to both recursive algorithms with applying the approximation via early stopping – described in Section 4.3. For the latter we use three different values for the stopping criterion γ , namely, 0.05, 0.10 and 0.20. We denote each of the above four runs for either problem by E, A_5, A_{10} and A_{20} , respectively.

Our goal is to evaluate the ability of our proposed approximations to capture the landscape of nondominated paths. When the shortest and the safest paths coincide (i.e. $|\mathcal{S}| = 1$) or when there are no intermediate paths between P_ℓ^* and P_r^* (i.e. $|\mathcal{S}| = 2$), the

Table 1: Summary of the solution sizes. We indicate the total number of solutions, number of singletons and of two-paths solutions ($|\mathcal{S}| = 1$ and $|\mathcal{S}| = 2$) and number of approximations multi-paths solutions identical to the exact solution.

	TOTAL-PATHS		MAX-PATHS	
	Chicago	Philadelphia	Chicago	Philadelphia
Total	2400	2400	2400	2400
$ \mathcal{S} = 1$	111	407	263	562
$ \mathcal{S} = 2$	182	148	317	253
Ident. A_5	520	476	398	424
Ident. A_{10}	420	407	340	351
Ident. A_{20}	337	327	14	49

approximations trivially return the same solution as the exact algorithms. Solutions containing intermediate paths (i.e. $|\mathcal{S}| > 2$) are called multi-paths solutions. The counts of singletons and two-paths solutions found for both cities and both problems are reported in Table 5. In addition, we also report the number of multi-paths solutions found with the approximations that are identical to the exact solutions. For instance, with the stricter approximation A_5 for the TOTAL-PATHS problem in Chicago, $111 + 182 + 520 = 813$ solutions out of 2400 obtained are identical to the exact solutions, as compared to only 630 with the looser approximation A_{20} . The difference is even more pronounced for the MAX-PATHS problem.

Still, we would like to measure the quality of the approximations beyond strict equality of the solution sets. For this purpose, we compare the area associated with the solution obtained with an approximation, denoted as S_A , with the area associated with the solution returned by the exact algorithm, denoted as S_E . Note that since the approximate solution is a subset of the exact solution, we have $S_E/S_A \leq 1$ and the equality holds if and only if the solutions are identical. Thus, we use the ratio S_E/S_A to measure the quality of the approximations. It takes values in the unit interval, with values closer to one indicating better approximations.

The statistics of our results on Chicago, grouped by distance classes, are presented as box plots in Figures 5–6. In both figures, the three lines of plots depict from top to bottom the distribution of (i) area ratios (S_E/S_A), (ii) solution sizes ($|\mathcal{S}|$) and (iii) runtimes in seconds (T) across the 400 pairs from each distance class. Note the different scale used for the last column in each figure (for D_5).

Results on Philadelphia are similar and we leave them out because of space constraints. For the same problem, the discrepancies arising between the two cities are due to characteristics of their respective road networks and crime distributions. For instance, because of the Delaware river that runs through Philadelphia, the walking distance between nodes on either side of the river is much greater than their distance “as the crow flies”. Hence, the closer distance classes contain pairs for which the shortest path is much longer than expected. Besides, crime locations are more spatially clustered in Chicago (see Figure 2) so the KDE for the crime activity is less regular, yielding a greater variety of risk thresholds in the MAX-PATHS problem.

While remaining of high quality (average area ratios S_E/S_A almost always lie above 0.8), the approximations clearly deteriorate for larger distances and especially for A_{20} . This is due to the fact that the average number of paths in the solutions returned by the exact algorithms increases as distances expand while the number of paths in the solutions with approximations is capped. The number of paths per solution averages between 7.83 and 16.33 and reaches up to 144 for the exact algorithms, while approximate solutions

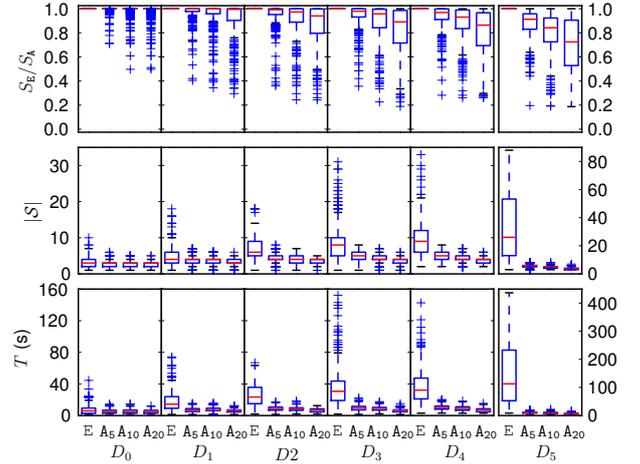


Figure 5: Results for TOTAL-PATHS on Chicago.

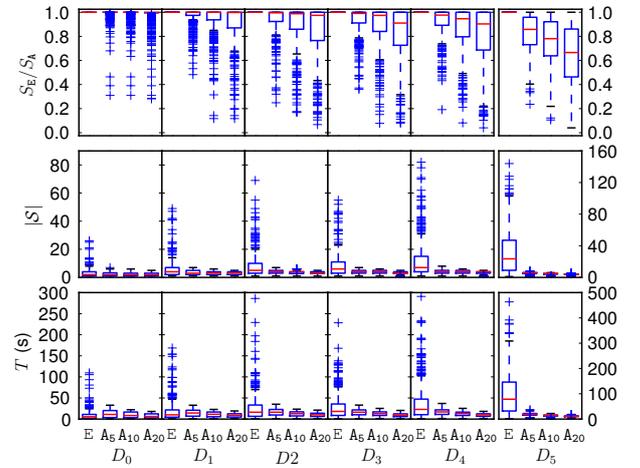


Figure 6: Results for MAX-PATHS on Chicago.

never contain more than 8 paths, with an average between 2.93 and 4.30 depending on the scenario. As expected, the runtime scales linearly with the size of the solution. Hence, combining the algorithms with approximations yields notable reductions in the runtimes, especially for larger distances.

In summary, by using moderate approximations (e.g. A_{10}) we obtain very satisfactory solutions containing a small number of paths, with shorter and more predictable runtimes, supporting the practical utility of our algorithms.

Pruning: Next, we investigate the benefits of applying the `Ellipse` pruning described in Section 4.3. These pruning techniques discard nodes that are too far away from s and t to be part of nondominating solutions. The first pruning technique, denoted by L_N , filters individual nodes while the second, denoted by L_G uses a grid and filters entire cells at once. Both techniques only affect the runtime of the algorithms without impacting the solutions.

For this experiment, we focus on the `Sum-Recursive` and `Max-Recursive` algorithms combined with approximation using $\gamma = 0.10$, as they represent the most practical scenario. The runtimes in seconds on the Chicago network for these algorithms with and without the pruning techniques are reported in Figure 5.

L_N drastically reduces the size of the input graph of the Dijkstra

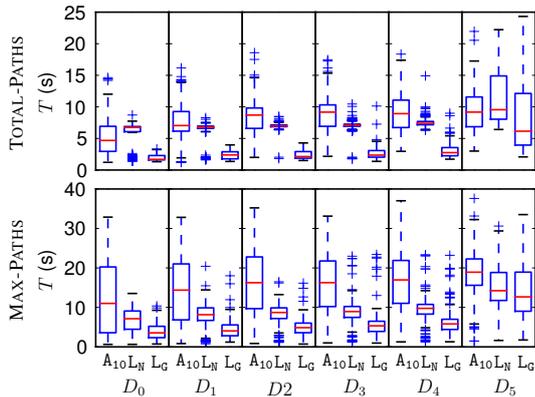


Figure 7: Runtime for pruning on Chicago.

routines, as witnessed by the very short span of runtimes. However, the pruning operation itself imposes a large overhead which takes away most of the benefits of using this method. This is particularly evident for the TOTAL-PATHS problem, where the average runtime with the pruning is almost equivalent to the average runtime of the original algorithm, if not greater.

On the contrary, L_G does not suffer from this drawback. While selecting whole cells at once is suboptimal in terms of the reduction of the graph, it can be performed more efficiently, resulting in an overall gain for both problems. Tuning the size of the grid provides a means to adjust the balance between stringency and speed of pruning. Here, after a short exploration, we chose a grid of 20×20 cells.

When the source and target nodes are far apart, the ellipse covers the graph almost entirely. Then, no node can be discarded and pruning only adds overhead runtime. However, such situations can be easily detected and the pruning can be abandoned.

6. RELATED WORK

Within the realm of urban computing [40], our work is related mainly to research on modeling crime events, on recommending routes to city-dwellers, and on solving biobjective optimization problems. Here, we review these lines of research and their connections to our study.

Crime event models: Crime events are neither random nor uniform [31], but they exhibit spatial and temporal patterns. Large research efforts focus on the development of specialized systems for *crime mapping* [26, 12, 33] that can be used by law enforcement agencies to study and analyze crime patterns and to inform their decisions on strategies against crime. While these methods can visualize events in the physical urban environment, there is also a large volume of urban and social literature that attempts to explain the reasons behind the observed patterns of crimes [11, 14, 15]. Such theories can also provide guidelines for policies that aim to prevent crimes. For instance, environmental criminology studies the relation between crime events and the spatio-temporal context of people’s activities [11]. Apart from these theories, there is a surge in studies that aim to identify and exploit patterns in historical crime data, hoping to forecast crimes (e.g. [36, 38, 25, 10]). These efforts have produced an arsenal of software tools that is available to law enforcement for proactive policing (e.g. COPLINK [21], ESRI’s GIS platform [6]).

While we are unable to present an exhaustive account of the cor-

responding literature, we emphasize here that our study is complementary in two ways: firstly, we propose computational methods relying on combinatorial algorithms that use crime data to recommend routes. Secondly, our algorithms could be combined with forecasting systems that project/update the probabilities of crime in the various street segments for upcoming path queries.

Urban navigation: Systems with new navigation objectives that go beyond the shortest path have recently appeared in the literature, considering the recommendation of healthy routes (i.e. maximizing physical activity) [34], “happy”, pleasant routes [29], routes tailored to the personal interests and spatiotemporal constraints of users [9, 27, 16, 18, 37], or routes that take into account traffic information gathered from historical speed patterns and GPS devices [19, 23, 39], among others. The PAROS interface [20] further allows to select routes based on up to three criteria, relying on the skyline algorithm proposed in [24]. To the best of our knowledge we are the first to consider the biobjective problem of safe and short routes.

Biobjective shortest paths: From a computational viewpoint, our work is related to research on finding non-dominated multi-objective paths, also known as skyline routes [24]. In particular, considering biobjective shortest paths problems, Shekelyan et al. [35] introduced the concept of linear paths skyline, that forms the lower convex hull of the nondominated solution points. They proposed refinements of the label-correcting algorithm, identified as the best solution strategy in an extensive comparison [30]. The algorithm we use requires less book keeping and was proposed by Mehlhorn and Ziegelmann [28] as part of their solution to a problem known as the *resource constrained shortest path*. It draws on geometric intuitions and lends itself to further approximations and speed-ups.

On the other hand, the MAX-PATHS problem proves to have a much more structured solution space, allowing us to identify complete sets of nondominated paths.

7. CONCLUSIONS

In this paper, we developed a risk model for an urban road network that is based on civic datasets of criminal activity as well as on city-dwellers mobility traces. More specifically, we used publicly available records of crime incidents from Chicago and Philadelphia as well as the urban road networks of these two cities obtained from OpenStreetMap.

Then we formalized the SAFEPATHS problem as a biobjective optimization problem and set as our goal to identify a small collection of nondominated paths that provide different tradeoffs between distance and safety. Our results showed the efficacy and the efficiency of our algorithms and demonstrated the practicality of biobjective optimization formulations in this setting.

By bringing together urban modeling based on public data sources and algorithmic solutions, we obtain an application that help city-dwellers to navigate safely their urban environment. In the future we plan to identify appropriate data sources that can drive the more accurate risk model described by Equation (2) as well as to incorporate the temporal dimension in our evaluations.

8. ACKNOWLEDGEMENTS

This work has been supported by NSF Awards: Award III # 1218437, CAREER Award # 1253393, IIS # 1320542, the Hariri Institute for computing, and gifts from Microsoft and Google.

9. REFERENCES

- [1] Crimes chicago - data portal:

- <https://data.cityofchicago.org/public-safety/crimes-2001-to-present/ijzp-q8t2>.
- [2] Crimes philadelphia - open data: <http://www.opendataphilly.org/opendata/resource/215/philadelphia-police-part-one-crime-incidents>.
 - [3] Openstreetmap: <http://www.openstreetmap.org>.
 - [4] Osm metro extracts: <http://metro.teczno.com>.
 - [5] `osm4routing` github: <https://github.com/tristramg/osm4routing>.
 - [6] Predicting crime using analytics and big data: <http://tinyurl.com/pjg3v68>.
 - [7] President Obama's administration-open government initiative: <http://www.data.gov/about>.
 - [8] United nations-world urbanization prospects: The 2011 revision - highlights: <http://esa.un.org/unup>.
 - [9] I. Ayala, L. Mandow, M. Amor, and L. Fuentes. An evaluation of multiobjective urban tourist route planning with mobile devices. In *LNCS Ubiquitous Computing and Ambient Intelligence, Vol 7656*, pp 387-394, 2012.
 - [10] R. Berk and J. Bleich. Statistical procedures for forecasting criminal behavior: A comparative assessment. In *Criminology and Public Policy 12 (3)*, 513-544, 2013.
 - [11] P. Brantingham and P. Brantingham. *Patterns in Crime*. Macmillan Publishing Company, 1984.
 - [12] S. Chainey and J. Ratcliffe. *GIS and Crime Mapping*. Wiley, ISBN 0-470-86099-5, 2005.
 - [13] S. Chainey, L. Tompson, and S. Uhlig. The utility of hotspot mapping for predicting spatial patterns of crime. In *Security Journal 21*, p. 4-28, 2008.
 - [14] L. Cohen and M. Felson. Social change and crime rate trends: a routing activity approach. *Am Soc Rev* 44:588-608, 1979.
 - [15] D. Cornish and R. Clarke. *The Reasoning Criminal: Rational Choice Perspectives on Offending*. 1986.
 - [16] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *HT*, 2010.
 - [17] I. Diakonikolas and M. Yannakakis. Small approximate pareto sets for biobjective shortest paths and other problems. *SIAM J. Comput.*, 39(4):1340-1371, 2009.
 - [18] A. Gionis, T. Lappas, K. Pelechrinis, and E. Terzi. Customized tour recommendations in urban areas. In *ACM WSDM*, 2014.
 - [19] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *VLDB*, 2007.
 - [20] F. Graf, H.-P. Kriegel, M. Renz, and M. Schubert. PAROS: Pareto Optimal Route Selection. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 1199-1202, New York, NY, USA, 2010. ACM.
 - [21] R. Hauck, H. Atabakhsb, P. Ongvasith, H. Gupta, and H. Chen. Using coplink to analyze criminal-justice data. In *Computer 35(3)*, 30-37, 2002.
 - [22] B. Hillier and O. Sahbaz. Beyond hot spots; using space syntax to understand dispersed patterns of crime risk in the built environment. In *Conference on crime analysis at the Institute of Pure and Applied Mathematics, University of California at Los Angeles*, 2007.
 - [23] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *IEEE ICDE*, 2006.
 - [24] H.-P. Kriegel, M. Renz, and M. Schubert. Route skyline queries: A multi-preference path planning approach. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 261-272, Mar. 2010.
 - [25] Y. Liu, M. Yang, M. Ramsay, X. Li, and J. Cold. A comparison of logistic regression, classification and regression tree, and neural network model in predicting violent re-offending. In *Journal of Quantitative Criminology*, 27:547-573, 2011.
 - [26] M. Maltz, A. Gordon, and W. Friedman. *Mapping Crime in Its Community Setting: Event Geography Analysis*. Springer-Verlag. ISBN 0-387-97381-8, 2000.
 - [27] A. Maruyama, N. Shibata, Y. Murata, K. Yasumoto, and M. Ito. P-tour: A personal navigation system for tourism. In *11th World Congress on ITS*, pp. 18-21, 2004.
 - [28] K. Mehlhorn and M. Ziegelmann. Resource Constrained Shortest Paths. In M. Paterson, editor, *Algorithms - ESA 2000*, volume 1879 of *Lecture Notes in Computer Science*, pages 326-337. Springer Berlin Heidelberg, 2000.
 - [29] D. Quercia, R. Schifanella, and L. Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *ACM Hypertext*, 2014.
 - [30] A. Raith and M. Ehr Gott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4):1299-1331, Apr. 2009.
 - [31] J. Ratcliffe. *Crime Mapping: Spatial and Temporal Challenges*. Springer Science and Business Media (Chapter 2 of *Handbook of Quantitative Criminology*), 2010.
 - [32] D. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, New York, Chichester, 1992.
 - [33] L. Scott and N. Warmerdam. Extend crime analysis with arcgis spatial statistics tools. In *ArcUser Magazine*, 2013.
 - [34] M. Sharkar, H. Karimi, and J. Zgibor. Health-optimal routing in pedestrian navigation services. In *ACM SIGSPATIAL HealthGIS*, 2012.
 - [35] M. Shekelyan, G. Jossé, M. Schubert, and H.-P. Kriegel. Linear Path Skyline Computation in Bicriteria Networks. In *Database Systems for Advanced Applications*, volume 8421, pages 173-187. 2014.
 - [36] T. Wang, C. Rudin, D. Wagner, and R. Sevieri. Learning to detect patterns of crime. In *ECML/PKDD*, 2013.
 - [37] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Social itinerary recommendation from user-generated digital trails. In *Personal and Ubiquitous Computing 16:469-484*, 2012.
 - [38] C. Yu, W. Ding, P. Chen, and M. Morabito. Crime forecasting using spatio-temporal pattern with ensemble learning. In *PAKDD*, 2014.
 - [39] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-drive: Enhancing driving directions with taxi drivers' intelligence. In *IEEE TKDE*, 2012.
 - [40] Y. Zheng, L. Capra, O. Wolfson, and Y. Hai. Urban Computing: Concepts, Methodologies, and Applications. *ACM Transaction on Intelligent Systems and Technology*, 2014.