Organizing Hot-Spot Police Patrol Routes

Sudarshan S. Chawathe

Computer Science Department
University of Maine
Orono, Maine 04469–5752, USA
chaw@cs.umaine.edu

***Abstract*—We address the problem of planning patrol routes to maximize coverage of important locations (hot spots) at minimum cost (length of patrol route). We model a road network using an edge-weighted graph in which edges represent streets, vertices represent intersections, and weights represent importance of the corresponding streets. We describe efficient methods that use this input to determine the most important patrol routes. In addition to the importance of streets (edge weights), important routes are affected by the topology of the road network. Our methods permit automation of a labor-intensive stage of the patrol-planning process and aid dynamic adjustment of patrol routes in response to changes in the input graph (as a result of a developing situation, for instance).**

I. Introduction

We address the task of planning patrol routes for crime prevention and response. Although, for concreteness, we present our work in the context of routes for police patrols on a road network, our methods are applicable to many other environments that call for the selection of surveillance and situation-response routes to maximize the benefit of the selected routes. Consider a network of roads, such as that found in a city, neighborhood, or other jurisdiction. Fig. 1 depicts a typical example, based on data from the *MassGIS* database [24]. We model this network using a graph in which vertices represent intersections and edges represent segments of streets between these intersections. Henceforth, we shall use the term street to mean street-segment, i.e., the portion of a street between two consecutive intersections. In a typical jurisdiction, we may expect several hundred edges in the graph representing the road network. Prior work has highlighted the importance of focusing on the spatial distribution of crime [31], [30]. In this paper, we model locations on a street-wise basis. That is, the importance of a location is quantified by assigning scores to the nearby streets (weights on the corresponding edges of the graph).

At first glance, it may seem that all we need to do to maximize benefit is to plan patrols that visit locations with frequencies that are proportional to their importance scores. However, this scheme works only if we ignore the topology of the road network. Equivalently, it works only if we assume that there is no cost associated with a patrol car moving from one location to another. While this assumption may be reasonable in extremely small jurisdictions, it does not hold true in typical jurisdictions consisting of hundreds or thousands of street-segments. For example, consider the small network of roads suggested by Fig. 6. If we pick the three most important locations (highest edge-weights), we obtain a patrol route composed of edges (b, f), (g, i), and (a, b). However, a transition between (g, i) and the other two edges requires traversing (b, g), and the cost of this traversal (travel time, work hours, etc.) cannot be avoided in this route. It may be better to select (b, g) instead of (a, b) even though the latter has a higher weight. Such effects of the topology of the road network rapidly increase in complexity as the network grows in size. In general, we wish to find *important patrol routes* (as an extension of important locations), which we define as those that provide the maximum benefit per unit cost. The benefit of a route is the sum of the importance scores of locations it visits (sum of the edge weights), while its cost is its length (number of edges).

We note that this paper addresses only one component of the complex process of planning patrol routes. Our goal is not to provide an automated replacement for the entire process. Rather, we propose a tool that automates a small but important and labor-intensive component of the overall process: determining important patrol routes based on importance scores of locations and the topology of the road network. This component depends crucially on other tasks, not addressed in this paper, on both its input and output side. On the input side, we do not address how importance scores are derived. (The road network itself, without the scores, is easily derived from existing maps and other planning data.) On the output side, we do not address how the proposed patrol routes are staffed, or how they are modified based on the expertise of the practitioners. This high-level data flow is summarized by Fig. 2. Important locations and their scores are determined using a combination of domain expertise, feedback from patrols, and recent events. Together with the road network, the scores define the weighted graph, which is the main input to our method. (The location tree is described in the next section.) Our hope is that an automated solution to part of the process will allow practitioners to better focus on the rest of the process. Our approach also permits some strategies that are otherwise impractical. For instance, it is possible to quickly recompute the potentially effective patrol routes when the importance scores of locations change as a result of ongoing activity.

1-4244-1330-3/07/$25.00 ©2007 IEEE.

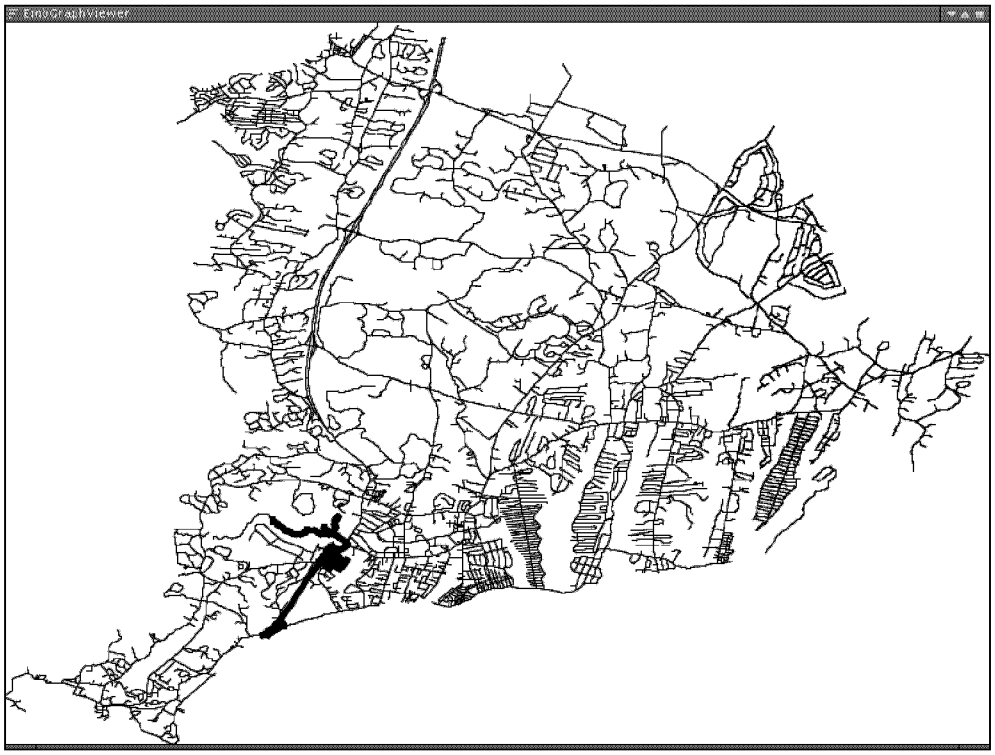

Fig. 1. A screenshot of the interactive visualization component of our patrol-planning system. The window shows a map of streets in Falmouth, Massachusetts.

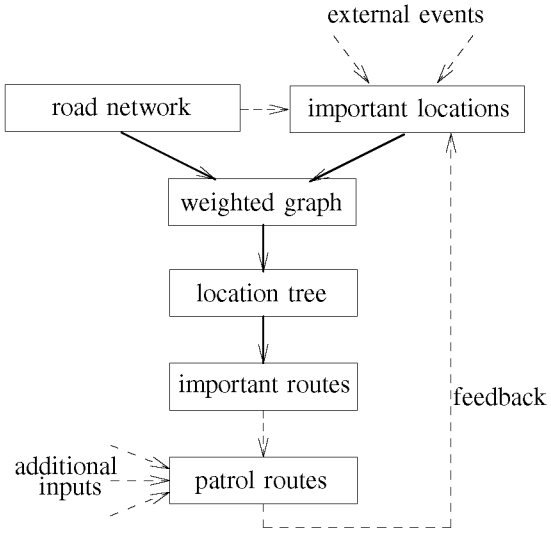

Fig. 2. High-level data-flow for patrol route planning. Solid arrows represent automated tasks addressed by this paper, while dashed arrows represent additional work, typically with human effort.

In our graph model, the benefit of a patrol route is the sum of the weights of its constituent edges, while its cost is the number of edges. We refer to their ratio, benefit/cost, as the *density* of the route or path. We need to specify two additional parameters to complete the problem definition: The first parameter, P, denotes the desired minimum length of a patrol route. If we do not specify a minimum length, then a route composed of only the heaviest edge is the densest because adding any additional edges will lower the average weight of edges in the route, and thus the density. More important, such short routes are not of much practical use for patrols. A typical value for P may range in the tens to hundreds, depending on factors such as the length of the street segment represented by an edge in the graph and the expected speed of travel. The second parameter, R, denotes the desired number of important routes. The simplest case is $R = 1$, when we find only the most important route. However, as noted above, there are several factors that are not modeled by our method that must be used to determine the final set of patrol routes. It is therefore useful to produce several important routes (say, a few dozen), in descending order of importance. One may expect here a parameter P' denoting the desired maximum length of a patrol route. This parameter is not as important as P, because longer routes tend to be not as dense as shorter ones. (In particular, if we are looking for important routes of length at least P, then there is always a densest—i.e., most important—route of length no greater than $2P - 1$.) We therefore skip the parameter P' in our descriptions below. However, it can be included by making a very simple change to our method.

After covering some preliminary ideas inn the next section, we present, in Section III, our method for computing high-density (i.e., important) routes based on a simple dynamic programming algorithm that operates on a spanning tree of the

graph modeling the road network. On a graph with N vertices and M edges, its running time is $O(M + N \log N + PRN)$ and its space requirement is $O(M+NP)$. We describe related work in Section IV and conclude in Section V.

II. Preliminaries

Recall, from Section I, that we model a network of roads using a graph, called the *street-graph*, in which vertices represent intersections and edges represent street-segments. The benefit of patrolling a street-segment is modeled by associating a weight, proportional to that benefit, with each edge. We do not model the cost of patrolling a street-segment explicitly but instead assume that the weights represent benefit per unit cost. For example, if the cost of street-segments is their length (in real distance on the ground), then the weight we associate with each edge is the ratio of that edge's benefit to its length. A patrol route is modeled as a *walk* (see below) in the street graph. The notion of the benefit of a street-segment is extended to that of a patrol in the natural manner: It is the sum of the sequence of weights corresponding to the sequence of edges forming the walk. We elaborate on these ideas below.

Following standard terminology [4], we define a *graph* $G = (V, E)$ where V is the set of *vertices* and $E \subseteq V^2$ is the set of *edges*. An edge $e = (v_1, v_2)$ is said to be *incident* on the vertices v_1 and v_2, which are called the *endvertices* of e. A *walk* of *length* k is an alternating sequence of $k + 1$ vertices and k edges, $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ such that each edge is incident on the vertices on either side of it; that is, v_{i-1} and v_i are the endvertices of e_i for $i = 1, \ldots, k$. The edges e_i and vertices v_i are not necessarily distinct. If $v_0 = v_k$, we say the walk is *closed*. We shall use the term *k-walk* to refer to a walk of length k. An *edge-weighted graph* is a graph $G = (V, E)$ along with an edge-weight function $w : E \to \mathbb{R}$.

The weight of a walk $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$ is $w(W) = \sum_{i=1}^{k} w(e_i)$. Note that when an edge e appears multiple times in a walk, it contributes an amount equal to its weight for each occurrence. In other words, a patrol route that covers a street multiple times yields a benefit for that street each time the street is traversed. This model does not distinguish between patrol routes that traverse a street multiple times in rapid succession and those that do so separated by larger time intervals. Consider the street-graph suggested by Fig. 3, with all edge weights being equal. Our definition of the weight (hence, benefit) of a walk yields the same weight for the following walks:

$$W_1 = a, 1, b, 2, c, 3, d, 4, e, 5, f, 6, a$$
$$W_2 = a, 1, b, 2, c, 3, d, 3, c, 2, b, 1, a$$

Intuitively, we may prefer W_1 over W_2 because it covers a larger number of streets. Such effects are not explicitly captured by our model.

Our focus is on determining patrol routes that provide the maximum benefit per unit cost, subject to the topological constraints of the street network.

We define the *density* $d(W)$ of a k-walk $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, with $k > 0$, to be its total weight divided by its length:

$$d(W) = \frac{1}{k} \sum_{i=1}^{k} w(e_i)$$

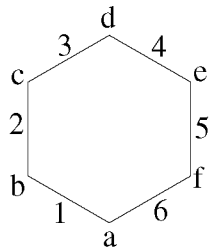

Fig. 3. A simple street-graph.

We may then state our problem as that of finding the densest walks of length at least K. We note that the phrase K-walk in the name of the problem statement below is interpreted as a walk of length at least K.

> *Densest Closed K-Walk (DCKW)*: Given an edge-weighted graph $G = (V, E)$ and a positive integer K, find a closed k-walk W in G, with $k \geq K$, such that there is no closed k'-walk in G, with $k' \geq K$, that is denser than W; that is, there is no closed k'-walk W' such that $k' \geq K$ and $d(W') > d(W)$.

Our focus on density rather than total weight is motivated by the need to balance the costs and benefits of patrols. While longer patrol routes may provide a larger benefit, the additional cost may not be justifiable. Even if the cost of traversing a street is incorporated into the weight of the corresponding edge in the graph (for instance, by setting the weight to the benefit per unit cost), computing only the walks of highest total weight is not a desirable strategy because the time spent on patrolling the less dense segments may be better spent (in terms of overall benefit) on patrolling the denser segments more frequently. Computing walks of large total weight also requires that the walks be suitably restricted because it is otherwise possible to extend any finite walk by another edge to obtain a walk of larger weight. One strategy is to limit attention *simple* walks, which are walks that to not repeat any vertex or edge, except that the first and last vertex may be the same. The problem is then a variant of the well-studied NP-hard problem of finding the longest path in a graph [18], [11], [19]. Other variations, such as one that requires all streets in a given set to be patrolled, at minimum cost, are very similar to other well-studied problems, such as the Chinese Postman Problem [25]. Although such variations are worth further study, we do not discuss them further in this paper, limiting ourself to our formulation that uses densest walks.

In general, there may be several walks in a graph satisfying the criteria for a densest closed K-walk (for a given K). For example, Fig. 4 suggests a graph, with all edge-weights set to one, in which each of the nested polygons is a densest 3-walk, with density equal to one.

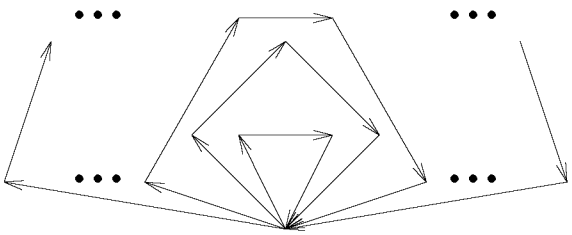

Fig. 4. A construction illustrating densest K-walks of multiple lengths.

In light of the above example, we may refine our problem definition by defining the *shortest* densest closed K-walk as follows. (As in our earlier problem definition, the phrase K-walk is interpreted as a walk of length at least K.)

> *Shortest Densest Closed K-Walk (SDCKW)*: Given an edge-weighted graph $G = (V, E)$ and a positive integer K, find a closed k-walk W in G, with $k \geq K$, such that
>
> - there is no closed k'-walk in G, with $k' \geq K$, that is denser than W; that is, there is no closed k'-walk W' such that $k' \geq K$ and $d(W') > d(W)$; and
> - all closed walks in G that are as dense as W are no shorter than W; that is, if W'' is a closed k''-walk with $k'' \geq K$ and $d(W'') = d(W)$ then $k'' \geq k$.

A graph may contain multiple shortest densest closed K-walks. An enumeration of all such walks is useful for patrol planning because we may then select among the alternatives, or combine some of them, based on other criteria for patrols. In general, it may also be desirable to extend such an enumeration to all densest walks (not just the shortest), or to walks within some threshold of the densest. However, the number of densest closed K-walks may be exponential in the size of the input graph, as suggested by the construction of . Fig. 5, with all edge weights being the same.

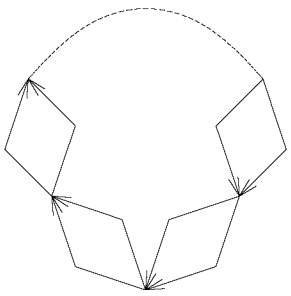

Fig. 5. A graph with an exponentially large number of densest closed walks.

In some instances, we may prefer longer walks to shorter ones. For example, longer patrols may be desirable as a means of covering a larger area using a small number of personnel. Now, any closed walk can be extended indefinitely by simply repeating the closed walk multiple times. Although such patrols may in fact be exactly what is desired, there is no additional practical value gained from such repeating or redundant closed walks.

In the next section, we focus on a restricted version of the shortest densest closed K-walk problem described above in which walks are required to traverse each used street once in each direction. That is, if a patrol route traverses a street in one direction then it must also traverse it in the reverse direction. Although this restriction does preclude some desirable patrol routes, the routes that remain are practically useful and can be computed very efficiently.

III. Important Patrol Routes

Recall that we quantify the importance of a route by its density, which is the ratio of its total weight to its length. Our task, then, is to find high-density paths in the given weighted graph. A naive method that computes the density of all paths, or an indiscriminate number of paths, is not feasible for even moderately large inputs because of the number of paths grows exponentially in the input size. By using methods for finding dense segments in sequences, such as the algorithm by Goldwasser, Kao, and Lu [13], it is possible to restrict our attention to only maximal paths in the graph, i.e., paths that cannot be extended with additional edges. However, the reduction in the number of paths to be considered is not sufficient and the resulting method remains impractical.

A. Location Tree

Our method for finding important patrol routes (densest routes) begins by computing a minimum spanning tree of the input graph, after temporarily reversing the sense of the edge weights, so that we obtain a heaviest spanning tree, which we call the *location tree*. During this step, the weight of each non-tree edge is added to the heaviest of the tree edges with which it shares a vertex. Although this transformation results in some loss of information, it also allows us to use a very efficient method to find the densest routes. Recall, from Section I, that the routes produced by our method are meant to serve as a basis for further analysis that incorporates additional factors. Furthermore, the inputs to the problem (importance scores of locations) are, by nature, imprecise. Thus it is reasonable to trade off accuracy for efficiency. Using standard methods (Prim's algorithm with a Fibonacci heap) [10], we can build the location tree in $O(M + N \log N)$ time; we therefore omit further details of this step in this paper. Fig. 6 suggests a sample location tree. This tree, and the rest of the example described below is artificially small for presentation purposes. As we have noted, a typical input is likely to contain hundreds or thousands of edges. Our methods are efficient enough for such inputs.

B. Densest Paths

Consider the set S_{np} of all paths that originate at a vertex n and contain exactly p edges. Let $X[n, p]$ be the density of the densest path in S_{np}, if $|S_{np}| > 0$, and 0 otherwise. Similarly, let $Y[n, p]$ be the density of the second-densest path in S_{np}, if $|S_{np}| > 1$, and 0 otherwise. In case of multiple densest paths in

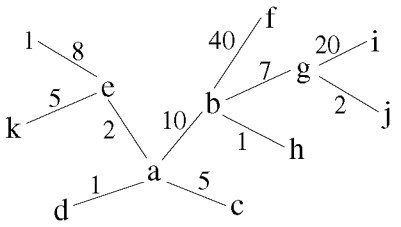

Fig. 6. A small example of a location tree. Edges represent streets and vertices, labeled using letters, represent intersections. The number next to each edge indicates its weight, i.e., importance.

S_{np}, we arbitrarily designate one as the densest and another as the second-densest. The case of multiple second-densest paths, but a single densest one, is handled analogously. Further, if $X[n,p] > 0$ then let $U[n,p]$ denote the vertex that immediately follows n in the densest path. Similarly, if $Y[n,p] > 0$ then let $V[n,p]$ denote the vertex that immediately follows n in the second-densest path.

We can compute the values $X[n,p]$ and $Y[n,p]$ (and $U[x,n]$ and $V[x,n]$) for all vertices n in a tree T, for increasing values of p, using a dynamic programming algorithm based on the following recurrence:

$$X[n,p] = \max_{(n,m)\in T} \left\{ \frac{w(n,m) + (p-1)\cdot Z(n,m,p-1)}{p} \right\} \quad (1)$$

where

$$Z(n,m,p-1) = \begin{cases} Y[m,p-1] & \text{if } U[m,p-1] = n \\ X[m,p-1] & \text{otherwise} \end{cases}$$

In either case, $U[n,p]$ is the m yielding the maximum value on the right-hand side of the equation. The recurrence for $Y[n,p]$ (and $V[n,p]$) is essentially as above, with the roles of X and Y, and U and V, interchanged, and with the max function replaced by one that picks the second-largest value.

Roughly, the recurrence says that the densest p-path (path of length p) originating from a vertex n begins with an edge e to a vertex m such that the sum of e's weight and the weight of the densest $(p-1)$-path originating at m is the largest among all neighbors m of n. However, we need to check for the case in which the densest $(p-1)$-path originating at m has (m,n) as the first edge. In this case, we must disregard the densest $(p-1)$-path from m and use the second-densest one instead. This condition is checked using the vertices stored in the U array.

C. Computing Densest Paths

The resulting algorithm is outlined in Listing 1. The function MPDENS computes the density of the densest paths in T that contain at least P edges. As suggested by line 36, it also saves the values of a few program variables that will be used by the function MDPATH, described below. Line 22 computes the fraction on the right-hand side of the recurrence above. The two-dimensional arrays X, Y, U, and V are populated in order of increasing p (second subscript) values. We note that the V array is not required for the computation and is included only for presentation purposes. The two outermost nested for-loops, on lines 10 and 11 iterate through the two dimensions of the arrays, corresponding to the two subscripts of the recurrence (1). The innermost loop, on line 16, iterates over all edges (n,m) incident on a vertex n, storing the highest and second-highest values of the expression in equation (1) along with the supplementary information in U and V.

Listing 1 Maximum path density in tree T

```
1:  function MPDENS(T, P)
2:      for all vertices n ∈ T do
3:          X[n,0] ← 0
4:          Y[n,0] ← 0
5:          U[n,0] ← n
6:          V[n,0] ← n
7:      end for
8:      d* ← 0
9:      p* ← 0
10:     for all p ← 1 ... 2P − 1 do
11:         for all vertices n ∈ T do
12:             X[n,p] ← 0
13:             Y[n,p] ← 0
14:             U[n,p] ← n
15:             V[n,p] ← n
16:             for all edges (n,m) ∈ T do
17:                 if U[m,p−1] = n then
18:                     Z ← Y[m,p−1]
19:                 else
20:                     Z ← X[m,p−1]
21:                 end if
22:                 d ← (w(n,m) + (p−1)·Z)/p
23:                 if d > X[n,p] then
24:                     Y[n,p] ← X[n,p]
25:                     V[n,p] ← U[n,p]
26:                     X[n,p] ← d
27:                     U[n,p] ← m
28:                     if p ≥ P ∧ d > d* then
29:                         d* ← d
30:                         p* ← p
31:                     end if
32:                 end if
33:             end for
34:         end for
35:     end for
36:     return d* (and save p*, X, U)
37: end function
```

The operation of the function MPDENS of Listing 1 on the sample data suggested by Fig. 6 is summarized in Fig. 7, for $P = 2$. Each set of four rows in the table (demarcated with horizontal lines) corresponds to one iteration of the for-loop on line 10 of Listing 1, for increasing values of the loop index p.

In Listing 1, the upper limit of the loop index p is set to $2P-1$. In case of an input parameter P' specifying the longest desired path, the upper limit may be replaced by P'. At this

p		vertex n a	b	c	d	e	f	g	h	i	j	k	l
0	$X[n,p]$	0	0	0	0	0	0	0	0	0	0	0	0
	$U[n,p]$	a	b	c	d	e	f	g	h	i	j	k	l
	$Y[n,p]$	0	0	0	0	0	0	0	0	0	0	0	0
	$V[n,p]$	a	b	c	d	e	f	g	h	i	j	k	l
1	$X[n,p]$	10	40	5	1	8	40	20	1	20	2	5	8
	$U[n,p]$	b	f	a	a	l	b	i	b	g	i	e	e
	$Y[n,p]$	5	10	0	0	5	0	7	0	2	0	0	0
	$V[n,p]$	c	a	c	d	k	f	b	h	j	j	k	l
2	$X[n,p]$	25	13.5	7.5	5.5	6	25	23.5	20.5	13.5	11	3.5	5
	$U[n,p]$	b	g	a	a	a	b	b	b	g	i	e	e
	$Y[n,p]$	5	7.5	0	0	0	0	11	0	0	0	0	0
	$V[n,p]$	e	a	c	d	e	f	i	h	i	j	k	l
3	$X[n,p]$	12.3	9.7	18.3	17	17.3	22.3	7.3	9.3	22.3	9.7	6.7	5.7
	$U[n,p]$	b	g	a	a	a	b	b	b	g	i	e	e
	$Y[n,p]$	0	6.7	0	0	0	0	0	0	0	0	0	0
	$V[n,p]$	a	a	c	d	e	f	g	h	i	j	k	l

Fig. 7. A summary of the operation of function MPDENS of Listing 1 on the location tree of Fig. 6, for $P = 2$.

point, the default choice of $2P - 1$ may seem curious. The reason for this choice is that there is always a path of length at most $2P - 1$ among all the densest paths of length at least P. To verify this claim, let R denote the set of densest paths and let r be a shortest path in R. If r were to contain $2P$ or more edges, then it would be possible to bisect r into two subpaths r_1 and r_2, each containing at least P edges. Further, the densities of r_1 and r_2 cannot both be lower than the density of r, so that we would arrive at a path that is shorter than r and at least as dense, which is a contradiction. It follows that there exists at least one densest path that contains fewer than $2P$ edges.

Listing 2 Recover maximum-density paths

```
1:  function MDPATH(T, d*, p*, X, U)
2:      R ← {()}
3:      for p ← p* ... 1 do
4:          for n ∈ T do
5:              if X[n,p] = d then
6:                  m ← U[n,p]
7:                  d ← (d · p − w(n,m))/(p − 1)
8:                  R' ← MDPATH(T, d, p − 1, X, U)
9:                  for r' ∈ R' do
10:                     R ← r'‖(n)
11:                 end for
12:             end if
13:         end for
14:     end for
15:     return R
16: end function
```

Function MPDENS computes only the density of the densest path, and not the path itself. The latter, however, is easily recovered from the saved X and U arrays as summarized in function MDPATHS, Listing 2. The main idea is to trace the evaluation of the recurrence (1) backwards by locating the vertex m that maximizes the right-hand side of the recurrence, as selected on line 27 of Listing 1. On line 10 of Listing 2, we use the symbol $\|$ to denote an operator that concatenates two sequences. The for loop on line 9 iterates over all sequences identified by the recursive invocation of the function. The for loop on line 4 finds all vertices that maximize the right-hand side of the recurrence. As a result, the function produces all the densest paths of length in $[P, 2P - 1]$. If only a single densest path is needed, the loops can be terminated when the first match is found.

As noted in Section I, it is useful to compute several, R, densest paths instead of only one or, as in Listing 2, only those of maximum density. For this purpose, we invoke MPDENS and MDPATH repeatedly on modified versions of the input. After each invocation, we modify T by setting to 0 the weights of all edges included in the output of MDPATH.

D. Analysis

The initial computation of the location tree has a space and time complexity of $O(M)$ and $O(M + N \log N)$, respectively, using standard methods [10]. The arrays X, Y, U, and V require space proportional to NP and all other program variable require only constant space. Thus the space complexity of MPDENS is $O(NP)$, for an overall space complexity of $O(M + NP)$. In each iteration of the for-loop on line 10 of Listing 1 (i.e., for each value of the loop index p), each edge of the location tree is examined once. Since the tree has $N-1$ edges, the loop requires $O(N)$ time, and the function overall requires $O(NP)$ time. When invoked $O(R)$ times to yield up to R important routes, the time complexity of the method overall is therefore $O(M + N \log N + PRN)$.

IV. Related Work

The importance of locations in understanding and controlling crime has been highlighted by much prior work, e.g., [31], [30]. The work described in this paper is relevant in the context of projects such as COPLINK [20]. For example, our method could be used to plan border patrols. In this context, work by Adam et al. on detecting anomalies based on data gathered from multiple sources is a potential source for the edge weights that form the input to our method [1]. Reis, Melo, Coelho, and Furtado present a genetic algorithm for simultaneously discovering crime hotspots and planning patrols [28]. That work shares with ours the goal of planning patrols. However, our approach is combinatorial instead of the genetic algorithm they employ. It should be interesting to combine the two approaches, especially in regard to discovering the edge weights used as the input to our method. Conversely, the important routes produced by our method could be used to speed up convergence of their method.

Our solution is based on simplifying the graph of the road network into a spanning tree with suitably adjusted edge weights. A much more extreme simplification of the problem restricts the graph to be a a single path of arbitrary length, i.e., a sequence. Although quite limiting, such a formulation has applications in planning highway patrols. Sections of a very long interstate highway, for example, may be reasonably modeled as a sequence by ignoring highway ramps and other side streets. In this case, it is possible to use work on the maximum-density segment problem. For example, Goldwasser, Kao, and Lu present a linear-time algorithm that uses a sweep-line data structure [13]. Lin, Huang, Jiang, and Chao present a simpler method for computing a set of densest non-overlapping segments based on maintaining a priority queue of subproblems generated by repeatedly splicing out dense segments [22].

Our method for computing important routes is based on the work of Lin, Kuo, and Chao on length-constrained maximum-density paths [21]. Similar methods have been applied to biomolecular sequence analysis [23]. It may be useful to generalize from important routes to important subtrees, or even important subgraphs. In the case of important subtrees, Hsieh and Chou present pseudo-polynomial-time algorithms for the finding the densest subtree [16]. In the case of general graphs, there is a significant amount of work on computing dense subgraphs under various constraints, where the density of a subgraph is defined as the ratio of the number of edges to the number vertices it contains [12], [6]. These notions of density are different from the one used in this paper and these results are not directly applicable to our problem. Nevertheless, it should be interesting to explore alternative formulations of the problem in the context of such work.

The model used in this paper does not make explicit use of the geometry of the street network. Rather, this model maps geometric aspects, such as the length of a street and its distance from an intersection, to the edge weights. Our focus has been on the constraints induced by the topology of the street network. In general, it is useful to consider both topological and geometric aspects of this problem. For example, it is useful to model the benefit of patrolling a street that is geometrically close to another, even if the latter street is not directly patrolled. It should be interesting to adapt spatial data structures and access methods for this purpose [15], [2], [29].

As noted in Section I, our methods are efficient enough to permit rapid recomputation of important patrol routes as conditions change. A related task is that of determining the current locations of patrol cars so that they may be efficiently rerouted as necessary. This problem is essentially that of map-matching: Given vehicle trajectory data (from on-board GPS units, dead-reckoning, and other sources) and a map, we need to determine the vehicle trajectory on the map. This problem is challenging because trajectory data is typically very noisy, with error margins larger than the distance between streets and other features. An incorrect decision at one point in the trajectory (such as picking the wrong option at a forking road) may result in amplified errors later in the trajectory (for instance, when the forked roads diverge to large distances). Several methods have been proposed for solving various versions of the map-matching problem [27], [5], [17], [3], [26], [14], [9], and those methods can be used in conjunction with the methods of this paper to provide a dynamic patrol-route guidance.

Our work in this paper has assumed that a single entity, such as a police jurisdiction, is the only party responsible for organizing patrols. It is often useful to model other parties who may have potentially conflicting interests in this regard. For example, the residents of a street may prefer frequent patrols on their street even if such patrols are not optimal in the sense of this paper. Given a fixed number of patrol officers, such desires of the residents of different streets are, in general, conflicting; that is, assigning more frequent patrols to one street is likely to require fewer patrols on others. The situation is complicated by the constraints induced by the topology of the network of streets. For example, a street that is on the most likely route from a central location to another street that has lobbied for additional patrols may be assured of frequent patrols. Such issues may be addressed using methods similar to those used for distributing costs over a network [8] and for determining fair traffic policies [7].

V. Conclusion

We have presented an efficient method for determining a set of important patrol routes. Our method aims to provide a tool that is part of a much larger system for planning patrols. The input to our method is an edge-weighted graph that represents a road network, with edge weights representing importance of the corresponding locations for patrolling purposes. In addition, our method takes as input a minimum and, optionally, maximum, route size, along with the number of desired routes. The output of our method is a set of routes that maximize the ratio of total edge weight to length, called the density. These routes are suitable for further analysis based on factors not

modeled by our method. An important feature of our method is its efficiency, which permits recomputation of important routes as changing conditions result in changes to the input edge weights. For example, changes due to a major accident or other problem can be quickly incorporated. Although, for concreteness, we presented our work in the context of a road network, the results have wider applicability. For example, we may use our methods for determining an efficient boarding schedule for transit police monitoring a network of subway trains, to coast-guard vessels patrolling shipping lanes in the vicinity of a busy harbor, and to hybrid networks composed of rail and road components.

In continuing work, we hope to extend the nature of both the input graph and the output routes. For example, we may wish to enumerate routes that are dense cycles instead of dense paths. We are also conducting an experimental evaluation of our method.

ACKNOWLEDGMENT

We thank an anonymous referee for insightful comments that led to improvements in presentation. This work is supported in part by the U.S. National Science Foundation with grant IIS-0081860 and CNS-0426683.

REFERENCES

[1] Nabil R. Adam, Vijayalakshmi Atluri, Rey Koslowski, Robert Grossman, Vandana P. Janeja, and Janice Warner. Secure interoperation for effective data mining in border control and homeland security applications. In *Proceedings of the International Conference on Digital Government Research (dg.o)*, pages 124–125, 2006.

[2] Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, Redondo Beach, California, November 2000.

[3] David Bernstein and Alain Kornhauser. An introduction to map matching for personal navigation assistants. Technical report, New Jersey TIDE Center, New Jersey Institute of Technology, Newark, New Jersey, August 1996.

[4] Béla Bollobás. *Modern Graph Theory*, volume 184 of *Graduate Texts in Mathematics*. Springer, New York, 1998.

[5] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 853–864, Trondheim, Norway, August 2005.

[6] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In Klaus Jansen and Samir Khuller, editors, *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 1913 of *Lecture Notes in Computer Science*, pages 84–95, Saarbrucken, Germany, September 2000. Springer.

[7] Sudarshan S. Chawathe. Fair policies for travel on neighborhood streets. In *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1027–1032, Vienna, Austria, September 2005.

[8] Sudarshan S. Chawathe. Distributing the cost of securing a transportation infrastructure. In *Proceedings of the 4th IEEE Intelligence and Security Informatics Conference (ISI)*, pages 596–601, San Diego, California, May 2006.

[9] W.C. Collier. In-vehicle route guidance systems using map matched dead reckoning. In *Proceedings of the IEEE Position Location and Navigation Symposium*, pages 359–363, 1990.

[10] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, June 1990.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman Company, November 1990.

[12] Andrew V. Goldberg. Finding a maximum density subgraph. Technical Report UCB/CSD 84/171, Computer Science Division, University of California, Berkeley, California, May 1984.

[13] Michael H. Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Fast algorithms for finding maximum-density segments of a sequence with applications to bioinformatics. In Roderic Guigó and Dan Gusfield, editors, *Proceeding of the 2nd International Workshop on Algorithms in Bioinformatics (WABI)*, volume 2452 of *Lecture Notes in Computer Science*, Rome, Italy, September 2002. Springer.

[14] Joshua Greenfeld. Matching GPS observations to locations on a digital map. In *Proceedings of the 81st Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2002.

[15] Erik G. Hoel and Hanan Samet. Efficient processing of spatial queries in line segment databases. In Oliver Günther and Hans-Jörg Schek, editors, *Proceedings of the 2nd International Symposium on Advances in Spatial Databases (SSD)*, volume 525 of *Lecture Notes in Computer Science*, pages 237–255, Zürich, Switzerland, August 1991. Springer.

[16] Sun-Yuan Hsieh and Ting-Yu Chou. Pseudo-polynomial time algorithms for the maximum-density subtree problem and related problems (extended abstract). In *Proceedings of the 23rd Workshop on Combinatorial Mathematics and Computation Theory*, pages 24–27, Changhua, Taiwan, April 2006.

[17] G. R. Jagadeesh, T. Srikanthan, and X. D. Zhang. A map matching method for GPS based real-time vehicle location. *The Journal of Navigation*, 57(3):429–440, September 2004.

[18] David Karger, Rajeev Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.

[19] Richard C. Larson and Amedeo R. Odoni. *Urban Operations Research*. Prentice-Hall, Englewood Cliffs, New Jersey, March 1981.

[20] Chienting Lin, Paul J. Hu, Hsinchun Chen, and Jennifer Schroeder. Technology implementation management in law enforcement: COPLINK system usability and user acceptance evaluations. In *Proceedings of the National Conference on Digital Government Research*, Boston, Massachusetts, May 2003.

[21] Rung-Ren Lin, Wen-Hsiung Kuo, and Kun-Mao Chao. Finding a length-constrained maximum-density path in a tree. *Journal of Combinatorial Optimization*, 9(1):145–156, 2005.

[22] Yaw-Ling Lin, Xiaoqiu Huyang, Tao Jiang, and Kun-Mao Chao. MAVG: locating non-overlapping maximum average segments in a given sequence. *Bioinformatics*, 19(1):151–152, 2003.

[23] Yaw-Ling Lin, Tao Jiang, and Kun-Mao Chao. Efficient algorithms for locating the length-constrainted heaviest segments, with applications to biomolecular sequence analysis. In Krzysztof Diks and Wojciech Rytter, editors, *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2420 of *Lecture Notes in Computer Science (LNSC)*, pages 459–470, Warsaw, Poland, August 2002. Springer.

[24] MassGIS database. Office of Geographic and Environmental Information (MassGIS), Commonwealth of Massachusetts Executive Office of Environmental Affairs. http://www.mass.gov/mgis/, December 2005.

[25] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, 1982.

[26] Mohammed A. Quddus. *High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications*. PhD thesis, Imperial College, London, U.K., January 2006.

[27] Mohammed A. Quddus, Robert B. Noland, and Washington Y. Ochieng. The effects of navigation sensors and digital map quality on the performance of map matching algorithms. In *Proceedings of the 85th Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2006.

[28] Danilo Reis, Adriano Melo, André L. V. Coelho, and Vasco Furtado. Towards optimal police patrol routes with genetic algorithms. In *Proceedings of the 4th IEEE Intelligence and Security Informatics Conference (ISI)*, pages 485–491, San Diego, California, May 2006.

[29] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, San Francisco, California, 2006.

[30] L. Sherman, P. Gartin, and M. Buerger. Hot spots of predatory crime: Routine activities and the criminology of place. *Criminology*, 27:27–56, 1989.

[31] D. Weisburd, L. Maher, and L. Sherman. Contrasting crime general and crime specific thory: The case of hot spots of crime. *Advances in Criminological Theory*, 4:45–70, 1992.