Journal of Service Science and Management, **2012, 5, 241-248**
http://dx.doi.org/10.4236/jssm.2012.53029 Published Online September 2012 (http://www.SciRP.org/journal/jssm)

Efficient Routing of Emergency Vehicles under Uncertain Urban Traffic Conditions

Amir Elalouf

Department of Management, Bar-Ilan University, Ramat Gan, Israel.
Email: amir.elalouf@biu.ac.il

Received May 22nd, 2012; revised June 20th, 2012; accepted July 2nd, 2012

ABSTRACT

Emergency-vehicle drivers who aim to reach their destinations through the fastest possible routes cannot rely solely on expected average travel times. Instead, the drivers should combine this travel-time information with the characteristics of data variation and then select the best or optimal route. The problem can be formulated on a graph in which the origin point and destination point are given. To each arc in the graph a random variable is assigned, characterized by the expected time to traverse the arc and the variance of that time. The problem is then to minimize the total origin-destination expected time, subject to the constraint that the variance of the travel time does not exceed a given threshold. This paper proposes an exact pseudo-polynomial algorithm and an ε-approximation algorithm (so-called FPTAS) for this problem. The model and algorithms were tested using real-life data of travel times under uncertain urban traffic conditions and demonstrated favorable computational results.

Keywords: Fast Routing Algorithm; FPTAS; Dynamic Programming

1. Introduction

The public's concern for safety and improving quality of life has generated a need for improved service of numerous public-safety and transportation agencies. An important challenge in this regard is to improve transportation system management for emergency response.

Minimization of emergency *response time* is a key focus in endeavors to improve emergency transport systems. A rapid response to an emergency situation can prevent or minimize adverse outcomes such as fatalities or the loss of property. In many cases, emergency vehicles (e.g., medical ambulances, police and security cars, fire and rescue vehicles, hazardous materials trucks, and gas/ electricity repair vehicles) are exempt from conventional traffic laws so that they may reach their destinations as quickly as possible; for example, they may be permitted to drive through red lights or exceed the speed limit.

Efficient routing under uncertain traffic conditions can improve the performance of intelligent transportation systems, wherein real-time traffic information is available at emergency dispatch centers. This research aims to develop and enhance a real-time emergency response system that uses real-time travel time information to assist the dispatchers of emergency vehicles in assigning the vehicles to optimal routes. To this end, this paper proposes a mathematical model for a decision-making process in real-time route dispatching. The model is formulated as a constrained shortest-path problem that can help the dispatcher to make decisions. The problem is stochastic path planning since, in contrast to standard deterministic routing problem, it takes traffic variability into account. In addition, the computational efficiency (algorithm complexity) of the process is examined, because this computational time has a major effect on the model's applicability in real-time situations.

The dispatching model proposed herein uses a dynamic-programming shortest-path algorithm as the basis for a fully polynomial time approximation scheme (FPTAS). Although a number of routing models have been developed previously, the current model is substantially more flexible, incorporating a joint analysis of the expected route time and its variance. The model and algorithms were tested using real-life data of travel time under uncertain urban traffic conditions and demonstrated favorable computational results.

2. Literature Review

Emergency vehicle routing (EVR) has been an attractive subject for operations researchers for many years. Most studies on EVR systems focus on location, fleet size, and operations performance. The majority of EVR mathematical models can be categorized into the following groups:

Group one: Location of emergency vehicle stations or

individual emergency vehicles within a region, subject to some performance criteria such as response time [1];

Group two: Determination of the minimum number of emergency vehicles required to cover a given area;

Group three: Dispatching strategies and their influence on performance results and services for emergency victims.

Work on emergency vehicle dispatching has generally involved the use of three approaches: queuing, mathematical programming, and dynamic programming (DP). Among studies using queuing methods, Larson [2,3] used a hypercube queuing model as a tool for facility location and redistricting in urban emergency services. Yet the ability to develop analytical methods (mathematical programming and queuing models) for EVR systems is rather limited, because even sophisticated analytical models cannot solve real-world problems under uncertain traffic data.

Cooke and Halsey [4] were the first to deal with time-dependent shortest path algorithms. Their algorithm is based on Bellman's principle of optimality and dynamic programming. Starting from the destination node, the algorithm calculates the path operating backwards. Ziliaskopoulos and Mahmassani [5] extended Cooke and Halsey's algorithm to calculate the time-dependent shortest paths from all nodes in a network to a given destination node over a given time horizon in a network with time-dependent arc costs. Goldberg and Paz [6] formulated an optimization model that allowed for stochastic travel times, unequal emergency vehicle utilizations, various call types, and service times that depend on call location. Ben-Akiva *et al.* [7] have shown that the static model outperforms the dynamic traffic assignment model for shorter prediction horizons. Hadas and Ceder [8] suggest a stochastic model for EVR under uncertainty that yields a set of paths and determines the probability that a given path is the shortest. The EVR algorithm proposed herein uses a combinatorial enumeration of prospective routes in combination with a simulation of several candidate shortest paths.

3. Problem Formulation

The problem framework is based on a computational network composed of a graph (possibly cyclic), $G = (N,A)$, with a set N of nodes, a set A of arcs, a start node $s = 1$, and a destination node $t = n$. The term t_{ij}, denoting the time to traverse arc (i,j) in G, is a random variable characterized by two parameters: m_{ij}—the expected time, and v_{ij} the variance. $|A| = h$, and $|N| = n$. The parameters m_{ij} are assumed to be integers. A path p is called *feasible* if the variance of the time to traverse the path is at most R, where R is a given threshold. The problem is to find a feasible path with minimum expected time.

Mathematical Form

Problem input: $G(N,A)$—a given graph.

For any arc $(i,j) \in A$, two parameters are given: m_{ij}—the expected time; v_{ij}—the variance.

$M(p)$ denotes the expected time to traverse path p;

$$M(p) = \sum_{(i,j)\in p} m_{ij}$$

$V(p)$ denotes the variance of the time it takes to traverse path p. All t_{ij} are assumed to be independent random variables and therefore $V(p) = \sum_{(i,j)\in p} v_{ij}$.

In a mathematical form the problem is to find a path p such that:

$$\min_p \left(M(p)\right)$$

s.t

$$V(p) \le R$$

4. Exact Algorithm: Dynamic Programming

This section introduces an exact DP algorithm. Since m_{ij} are assumed to be integers, DP is a pseudo-polynomial algorithm. As a result, if m_{ij} is bounded, DP is polynomial.

Let us associate with each path p a pair (M, V), where $M = M(p)$ is the expected time to traverse path p, and, correspondingly, $V = V(p)$ is the variance of the time to traverse p. Sets $S(k)$ of pairs (M, V) are arranged in increasing order of M-values, so that every pair in $S(k)$ corresponds to a path from node s to a node k. In order to restore the path corresponding to a pair (M, V), each pair is assigned a predecessor pair, and then standard backtracking can be used.

If there are two pairs in $S(k)$, $(M1,V1)$ and $(M2,V2)$ such that $M1 \le M2$ and $V1 \le V2$, then the pair $(M2, V2)$ is called *dominated* and may be discarded. Let UB be the upper bound of the total expected time for the optimal path. For instance, UB can be set to $\sum_{(i,j)\in A} m_{ij}$. The detailed polynomial time DP algorithm is presented in **Figure 1**.

Proposition 1. The complexity of the DP algorithm (Algorithm 1) is $O(hnUB)$.

Proof: Since the times are integers and dominated pairs are discarded, there are at most UB pairs in sets W and $S(k)$. Furthermore, construction of W in lines 9 - 11 requires $O(UB)$ elementary operations, since W is constructed from a single $S(k)$. Merging the sorted sets W and $S(k)$, in line 12, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is at most UB). In Step 2, lines 5 - 14 have two nested loops, where the first begins at line 6 and the second at line 7. These two loops go over all the arcs $n - 1$ times, so that in total, there are $O(hn)$ iterations of lines 11 - 13. Thus, the total complexity of Algorithm 1 is

Algorithm 1. The exact DP algorithm
1. Input: G(N,A) $|N| = n$, $|A| = m$, $\{(m(i,j), v(i,j)) | (i,j) \in A\}$; R
2. Output: A variance-constrained path with minimum expected time
3. **Step 1.** [Initialization]
4. Set $S(1) = \{(0,0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2 \cdots n$
5. **Step 2.** [Generate $S(2)$ to $S(n)$]
6. Repeat $n - 1$ times
7. for *each arc* $(u,k) \in A$ (*leading from node u to node k*)
8. $W \leftarrow \varnothing$
9. for each pair $(M,V) \in S(u)$
10. do if $V + v(u,k) \le R$ then $W \leftarrow W \cup \{(M + m(u,k), V + v(u,k))\}$
11. endfor
12. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs
13. endfor
14. End Repeat
15. **Step 3.** [Determine optimal solution]
16. find min M in $S(n)$, denote it by *ans*
17. Return *ans* as the optimal time; use backtracking to find optimal path.

Figure 1. The exact pseudo-polynomial DP algorithm.

O($hnUB$). □

5. Fully Polynomial Time Approximation Scheme

5.1. General Description of the FPTAS

Our approach for constructing an FPTAS follows a so-called *interval partitioning* computational scheme. The interval partitioning technique was originally proposed by Sahni [9] for the knapsack problem and was later improved by Gens and Levner [10] and by Levner *et al.* [11] and Elalouf *et al.* [12]. The FPTAS in this paper consists of three main stages:

Stage A: Find a preliminary lower bound LB and an upper bound UB on the optimal path's expected time such that $UB/LB \le n$.

Stage B: Find improved lower and upper bounds such that $UB/LB \le 2$.

Stage C: Partition the interval [LB,UB] into n/ε equal subintervals, delete sufficiently close solutions in the subintervals, and then find an ε-approximation solution using full enumeration of the "representatives" (taking only one "representative" from each subinterval).

This technique is similar to that presented by the author at the conference EOMAS-2011 and published in the conference proceedings [12]. Notice, however, that the present paper addresses another type of problem. Specifically, the underlying graph G is allowed to have cycles, and, in addition, the problem under consideration is a problem of minimizing, rather than maximizing, the objective function. As a result, the new algorithm has a different complexity.

5.2. Stage A: Finding Preliminary Lower and Upper Bounds

The following greedy technique is applied: Let $A = \{a_1, a_2, \cdots, a_m\}$ be the set of arcs in $G(N,A)$. Denote graph $G'(N',A')$ with the same set of nodes, $N' = N$, and the set of arcs $A' \subseteq A$. To define A', a binary variable, x_{ai} is used. If $x_{ai} = 1$ then $a_i \in A'$; otherwise $a_i \notin A'$. The arcs in G are ordered according to their non-decreasing expected time values: $m_{[a1]} \le m_{[a2]} \le \cdots \le m_{[ah]}$, and are initialized at $x_{ai} = 0$ for any $i = 1, \cdots, h$ (*i.e.*, the initial G' is a graph with no arcs). Then the x_{ai} s are iteratively set at $x_{[a1]} = 1, x_{[a2]} = 1, \cdots$ and each arc is added to the graph until a path is obtained from the source to the destination that satisfies the constraint. If all $x_{ai} = 1$ but such a path cannot be found, there is no feasible solution. Let x_k be the last variable set to 1 in the above procedure. Then set $m_0 = m_{ak}$. Obviously, the optimal total travel time (denoted by OPT) must lie between m_0 and nm_0. In cases where the OPT equals zero, the above greedy procedure in Stage A finds the exact optimal solution (*i.e.*, a path of zero duration); therefore, Stages B and C are not required.

Complexity of Stage A. Sorting the arcs is done in O(h log h). Each check of whether the graph G has a feasible path on a selected set of arcs requires O(n^2) time [13]. The total number of checks is O(log$h$) if a binary search is performed in the interval [1, h]. Thus, the complexity of Stage A here is O(n^2log$h$).□

5.3. Stage B: Finding Improved Bounds

This stage has two building blocks: a test procedure, denoted Test (w,ε), and a narrowing procedure denoted BOUNDS that uses Test (w,ε) as a sub-procedure.

5.3.1. Test Procedure (Test(w,ε))

Test(w,ε) is a parametric dynamic-programming type algorithm that has the following property: given positive parameters w and ε, it either reports that the minimum possible expected travel time $M^* \le w$, or, otherwise, reports that $M^* \ge w(1 - \varepsilon)$.

Test(w,ε) will be repeatedly applied as a sub-procedure in the algorithm BOUNDS below to narrow the gap between UB and LB until $UB/LB \le 2$.

Associate with each path p a pair (M,V), where $M = M(p)$ is the path expected travel time, and, correspondingly, $V = V(p)$, the variance of path time. Sets $S(k)$ of pairs (M,V) are arranged in increasing order of M-values so that every pair in $S(k)$ corresponds to a path from the start node s to a node k. As in DP, the dominated pairs in all $S(k)$ sets are deleted.

In addition to dominated pairs, the δ-close pairs are deleted as follows: Given a δ, if there are two pairs in $S(k)$, ($M1$, $V1$) and ($M2$, $V2$), such that $0 \le M2 - M1 \le \delta$, then the pairs are called δ-close. The δ-close pairs are discarded from set $S(k)$ according to the following procedure:

1) Let w be a given parameter satisfying $LB \le w \le UB$. In each $S(k)$, partition the interval $[0,w]$ into $\lceil n/\varepsilon \rceil$ equal subintervals of size no greater than $\delta = \varepsilon w/n$;

2) If more than one pair from $S(k)$ falls into any one of the above subintervals, discard all such δ-close pairs, leaving only one representative pair in each subinterval, namely the pair with the smallest (in this subinterval) V-coordinate.

3) A pair with (M,V) with $M > w$ (called w-redundant) may be discarded.

The algorithm for Test(w,ε) is presented in **Figure 2**.

Proposition 2. The complexity of Test(w,ε) is $O(hn^2/\varepsilon)$,

Proof: Since the subinterval length is $\delta = \varepsilon w/n$, there are $O(n/\varepsilon)$ subintervals in interval $[0,w]$. Therefore there are $O(n/\varepsilon)$ representative pairs in sets W and $S(k)$. Further, constructing each W in lines 10 - 12 requires $O(n/\varepsilon)$ elementary operations. Merging the sorted sets W and $S(k)$, in line 13, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is $O(n/\varepsilon)$). In Step 2 the algorithm goes over all the arcs n-1 times, so that in total, there are $O(nh)$ iterations of lines 10 - 12. Thus, the total complexity of Algorithm 2 is $O(hn^2/\varepsilon)$. □

Algorithm 2. The Testing Procedure (Test(w,ε))

1. Input: G(N,A) $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j)) | (i,j) \in A\}$; R
2. Input ε,w
3. $\delta \leftarrow \varepsilon w/n$
4. **Step 1.** [Initialization]
5. Set $S(1)=\{(0,0)\}$, $S(k)\leftarrow\varnothing$ for $k = 2 \cdots n$
6. **Step 2.** [Generate $S(1)$ to $S(n)$]
7. Repeat $n-1$ times
8. for *each arc* $(u,k) \in A$ (*leading from node u to node k*)
9. $W\leftarrow\varnothing$
10. for each pair $(M,V) \in S(u)$
11. do if $V + v(u,k) \le R$ then $W\leftarrow W \cup \{(M + m(u,k), V + v(u,k))\}$
12. endfor
13. $S(k)\leftarrow$merge($S(k), W$); during merging eliminate the dominated and δ-close pairs
14. endfor
15. End Repeat
16. **Step 3.** Find a pair (M,V) in $S(n)$, such that $M \le w$.
17. If such a path is found in $S(n)$, return $M^* \le w$.
18. If such a path cannot be found in $S(n)$ return $M^* \ge w(1-\varepsilon)$

Figure 2. The testing procedure test (v, ε).

5.3.2. The Narrowing Procedure (BOUNDS)

The narrowing procedure presented in this section (**Figure 3**) is adapted from the procedure suggested by Ergun *et al.* [14] for solving the restricted shortest path. Namely, when Test(w,ε) runs, the ε value is chosen as a function of UB/LB, and this value is updated from iteration to iteration. To distinguish the allowable error (ε) in the FPAS from the iteratively changing error in the testing procedure, the latter is denoted as θ. The complexity of BOUNDS is $O(hn^2)$. The proof is the same as that of Lemma 5 in [14].

5.4. Stage C: The ε-Approximation Algorithm

Stage C is initiated with LB and UB values satisfying $UB/LB \le 2$, and obtains an ε-approximation path.

Associate with each path p a pair (M,V), where $M = M(p)$ is the path expected time, and, correspondingly, $V = V(p)$ is the path variance. Sets $S(k)$ of pairs (M,V) are arranged in increasing order of M-values so that every pair in $S(k)$ corresponds to a path from the start node s to a node k. As in DP all the dominated pairs are deleted in all $S(k)$ sets.

In addition to dominated pairs, the δ-close pairs are deleted as follows: Given a δ, if there are two pairs in $S(k)$, ($M1$, $V1$) and ($M2$, $V2$), such that $0 \le M2 - M1 \le \delta$, then the pairs are called δ-close. The δ-close pairs are discarded from each set $S(k)$ according to the following procedure:

1) In each $S(k)$, partition the interval $[0,UB]$ into $\lceil (UB/LB)(n/\varepsilon) \rceil$ equal subintervals of size no greater than $\delta = \varepsilon LB/n$;

2) If more than one pair from $S(k)$ falls into any one of the above subintervals, discard all such δ-close pairs, leaving only one representative pair in each subinterval, namely the pair with the largest (in this subinterval) V-coordinate.

3) A pair (M,V) with $M > UB$ may be discarded.

The corresponding algorithm is presented in **Figure 4**.

Algorithm 3. BOUNDS

1. Input: LB and UB such that $UB/LB \le n$.
2. Output: LB and UB such that $UB/LB \le 2$
3. If $UB/LB \le 2$, Goto 10
4. Set $\theta \leftarrow \sqrt{UB/LB} - 1$
5. Set $w \leftarrow \sqrt{LB \cdot UB}/(1-\theta)$
6. Run Test(w, θ)
7. If Test(w, θ) returns that $M^* \le w$ then set $UB\leftarrow w$
8. else set $UB\leftarrow w(1-\theta)$
9. Go to line 3
10. Return the improved LB,UB
11. End

Figure 3. BOUNDS algorithm.

Algorithm 4.
The ε-approximation algorithm AA (LB, UB, ε)
1. Input: G(N,A) $|N| = n$, $|A| = h$,
   $\{(m(i,j), v(i,j)|(i,j) \in A\}$; R
2. Input UB, LB, ε
3. $\delta \leftarrow \varepsilon LB/n$
4. Output: ε-approximation path such that path expected time is at most $(1 + \varepsilon)$OPT
5. **Step 1.** [Initialization]
6. Set $S(1) = \{(0,0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2 \cdots n$
7. **Step 2.** [Generate $S(2)$ to $S(n)$]
8. Repeat $n - 1$ times
9. for *each arc* $(u,k) \in A$ (*leading from node u to node k*)
10. $W \leftarrow \varnothing$
11. for each pair $(M,V) \in S(u)$
12. do if $V + v(u,k) \le R$ then $W \leftarrow W \cup \{(M + m(u,k), V + v(u,k))\}$
13. endfor
14. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated and δ-close pairs
15. endfor
16. End Repeat
17. **Step 3.** [Determine approximate solution]
18. find min M in $S(n)$, denote it by *ans*
19. Return *ans* as the ε-approximation expected time, use backtracking to find the path
20. The path's expected time is at most $(1 + \varepsilon)$ OPT.

Figure 4. The ε-approximation algorithm AA (LB, UB, ε).

Theorem 1. The complexity of AA (LB, UB, ε) is $O(hn^2/\varepsilon)$. The complexity of the entire three-stage FPTAS is $O(hn^2/\varepsilon)$.

Proof: Since the subinterval length is $\delta = \varepsilon LB/n$, there are $O(n(UB/LB)(1/\varepsilon))$ subintervals in interval $[0,UB]$, and since $UB/LB \le 2$, it comes to $O(n/\varepsilon)$ subintervals in interval $[LB,UB]$. Therefore, there are $O(n/\varepsilon)$ representative pairs in any set W, T and $S(k)$. Constructing each W in lines 11 - 13 requires $O(n/\varepsilon)$ elementary operations, since W is constructed from a single $S(k)$. Merging the sorted sets W and T, in line 14, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is $O(n/\varepsilon)$). In Step 2 there are $O(nh)$ iterations of lines 11 - 13. Thus, Algorithm 4 has the complexity of $O(hn^2/\varepsilon)$ in total. Since step C dominates steps A and B of the algorithm, the complexity of the entire approximation algorithm is $O(hn^2/\varepsilon)$. □

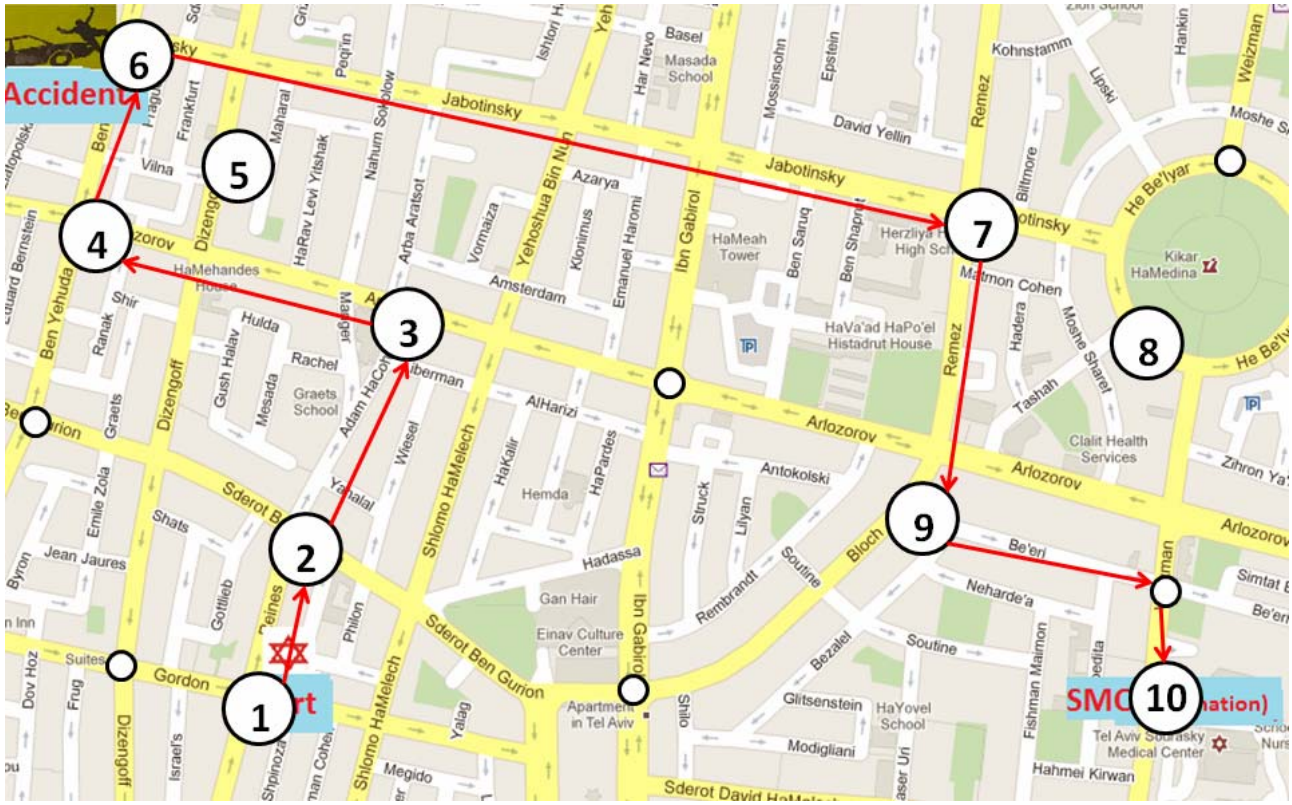
6. Example

This section presents a practical numerical example of an application of the model. **Figure 5(a)** shows a Google Map representation of a region in Tel Aviv. Assume that a car accident has occurred at the point denoted "Accident" on the map, and the nearest ambulance originates from the point denoted "Start". The destination point for transporting accident victims is the Tel Aviv Sourasky Medical Center, denoted SMC on the map. The ambulance driver needs to choose the route with minimum expected time, satisfying the constraint that the route time variance does not exceed a given threshold $R = 4$. The corresponding graph with 10 main crossroads and 23 main possible roads is presented in **Figure 6**. Real-life data are obtained by a statistical survey and presented in **Table 1**; all the numbers in the table are given in minutes. The last two lines are the average times and corresponding variances. The restricted shortest route found by our algorithm is shown in **Figure 5(b)**. Its total response time is 6.6 with a variance of 3.67, which is less than the allowed value of $R = 4$. Notice that the shortest unrestricted route in this problem has a total duration of 4.9 minutes,

Table 1. Point to point times (in minutes).

	1→2	2→3	2→4	3→4	3→5	3→6	5→6	6→7	7→8	7→9	8→9	8→10	9→10
1	0.20	0.83	3.99	1.18	0.25	1.84	1.57	0.77	0.54	0.63	3.47	1.75	0.82
2	0.26	0.85	4.27	1.14	0.21	1.29	2.26	0.76	0.64	0.17	3.85	0.11	1.01
3	2.53	0.79	3.61	1.02	0.26	1.34	2.71	0.54	2.50	0.66	4.84	0.82	0.77
4	0.31	0.73	3.80	1.30	0.21	1.27	2.06	0.24	0.29	1.90	2.58	0.50	1.07
5	0.26	0.91	3.25	1.56	0.40	0.44	2.08	0.59	0.41	0.52	2.24	0.56	1.15
6	0.24	0.68	3.21	1.47	0.15	0.34	1.69	1.09	0.70	0.20	3.40	0.76	0.97
7	0.31	0.76	2.93	1.30	0.20	1.25	2.20	1.60	0.61	0.93	0.60	0.56	0.78
8	0.32	0.78	3.39	0.99	0.20	1.07	2.42	0.81	0.80	0.35	3.35	1.57	1.12
9	0.33	0.80	3.63	1.60	0.20	0.98	1.60	0.36	0.50	1.98	2.22	0.52	0.93
10	0.51	0.75	3.18	1.13	3.50	1.10	2.10	1.09	0.37	0.53	2.07	1.44	0.99
Average	0.53	0.79	3.53	1.27	0.56	1.09	2.07	0.79	0.74	0.79	2.86	0.86	0.96
Data variance	0.71	0.07	0.41	0.22	1.04	0.44	0.37	0.40	0.64	0.65	1.18	0.54	0.14

(a)



(b)

Figure 5. (a) Accident area map (Google Maps); (b) A restricted shortest path found.

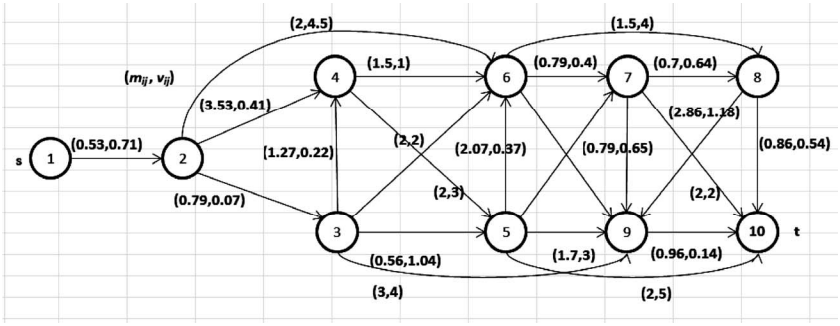


Figure 6. Graph corresponding to main roads.

but its variance is 9.75, which exceeds the given threshold, $R = 4$. In the worst case, the practical realization of this (random) path duration may be 15 minutes, which is unacceptable in practice.

7. Concluding Remarks

Emergency vehicles are essential for population security. They must respond to emergency situations as quickly as possible and operate under varying conditions. This study improves the state of the art in emergency vehicle routing by introducing stochastic path planning taking the traffic variability into account. The paper's results demonstrate that stochastic path planning can reduce emergency vehicle travel time in practice. Stochastic path planning shows significant benefits over static path planning. Thus, in situations in which traffic is intensive and highly variable, stochastic path planning can play a significant role in saving lives.

An efficient routing algorithm is an indispensable component of commercial transportation software such as Google Maps [15], GAMS_Transportation Problem Solver [16], and TransCAD [17]. More than that, the constrained routing problem considered in this paper is not only of interest on its own; actually, it arises as a subproblem in many more general optimization problems such as planning and scheduling problems [18], lot-sizing problems [19] and facility location problems [20]. The efficient algorithm for finding the fastest (or cheapest, most profitable, robust, etc.) route usually serves as a subroutine in more general algorithms developed for solving these problems.

This paper proposes an auxiliary dynamic programming approach for developing a fast routing strategy. Since for a large-scale network the complexity of this type of algorithm might be high, a fast approximation algorithm—based on a technique called interval partitioning [9] is introduced. This is the first study to use this approximation technique to solve a problem with uncertain data. To summarize, the mathematical model and algorithms presented in this paper can serve a basis for future real-time emergency vehicle dispatching systems. They have the following advantages when applied to real-time emergency vehicle routing dispatching:

- The emergency vehicle dispatching model can use real-time traffic information to dynamically compute the shortest paths. The model's ability to utilize real-time information can improve EVR performance greatly, especially when traffic is dense and there is significant congestion in the road networks.
- The model incorporates a flexible strategy and mathematical optimization using efficient enumeration of numerous routes. This model ensures that the EVR system works efficiently in most real-life situations.
- The system enables responders to handle emergency situations in a timely fashion, *i.e.*, with minimal response times. The proposed algorithm is capable of generating solutions in which travel time variance does not exceed a predetermined threshold; this property is especially important in urban areas with intensive and highly variable traffic. Notice that standard shortest path algorithms do not possess this property. An example of a practical application of the algorithm is given in the previous section.

Future research can expand this work in several directions. First, it would be of interest to develop more realistic model formulations that can handle other types of traffic constraints and additional delay factors encountered in the real world. Along with the typical constraint considered in this paper, according to which the variance of the path travel time cannot exceed a prescribed value, other types of probabilistic constraints are of interest; for instance, future models may pose constraints on the probability that path travel time exceeds a given value. A second potential focus for future work is to design new improved algorithms for solving these models. Furthermore, it would be beneficial both from theoretical and practical points of view to extend the exact and approximation algorithms proposed herein to solve more general lot-sizing and scheduling problems with stochastic parameters. These algorithms could subsequently be compared with the procedures available in existing comercial software systems.

Incorporation of the models and algorithms presented in this paper into a commercial vehicle dispatching system may improve system performance, and assessment of the real-world applicability of these algorithms constitutes another attractive direction for future research.

REFERENCES

[1] C. Toregas, R. Swain, C. ReVelle and L. Bergman, "The Location of Emergency Service Facilities," *Operations Research*, Vol. 19, No. 6, 1971, pp. 1363-1373. doi:10.1287/opre.19.6.1363

[2] R. Larson, "A Hypercube Queuing Model for Facility Location and Redistricting in Urban Emergency Services," *Computers and Operations Research*, Vol. 1, No. 1, 1974, pp. 67-95. doi:10.1016/0305-0548(74)90076-8

[3] R. Larson, "Approximating the Performance of Urban Emergency Service Systems," *Operations Research*, Vol. 23, No. 5, 1975, pp. 845-868. doi:10.1287/opre.23.5.845

[4] K. Cooke and E. Halsey, "The Shortest Route through a Network with Time-Dependent Internodal Transit Times," *Journal of Mathematical Analysis and Applications*, Vol. 14, No. 3, 1966, pp. 493-498. doi:10.1016/0022-247X(66)90009-6

[5] A. Ziliaskopoulos and H. Mahmassani, "Time Dependent, Shortest-Path Algorithm for Real-Time Intelligent Vehicle Highway System Applications," *Transportation Research Record*, Vol. 1408, 1993, pp. 94-100.

[6] J. Goldberg and L. Paz, "Locating Emergency Vehicle Bases When Service Time Depends on Call Location," *Transportation Science*, Vol. 28, No. 4, 1991, pp. 264-280. doi:10.1287/trsc.25.4.264

[7] M. Ben-Akiva, E. Cascetta and H. Gunn, "An On-Line Dynamic Traffic Prediction Model for an Inter-Urban Motorway Network," In: N. H. Gartner and G. Improta, Eds., *Urban Traffic Networks*: *Dynamic Flow Modeling and Control*, Springer, Berlin, 1995, pp. 83-122.

[8] Y. Hadas and A. Ceder, "Shortest Path of Emergency Vehicles under Uncertain Urban Traffic Conditions," *Transportation Research Record*, Vol. 1560, No. 1, 1996, pp. 34-39. doi:10.3141/1560-06

[9] S. Sahni, "Algorithms for Scheduling Independent Tasks," *Journal of the ACM*, Vol. 23, No. 1, 1976, pp. 116-127. doi:10.1145/321921.321934

[10] G. V. Gens and E. V. Levner, "Fast Approximation Algorithms for Job Sequencing with Deadlines," *Discrete Applied Mathematics*, Vol. 3, No. 4, 1981, pp. 313-318. doi:10.1016/0166-218X(81)90008-1

[11] E. Levner, A. Elalouf and T. C. E. Cheng, "An Improved FPTAS for Mobile Agent Routing with Time Constraints," *Journal of Universal Computer Science*, Vol. 17, No. 13, 2011, pp. 1854-1862.

[12] A. Elalouf, E. Levner and T.C.E Cheng, "Efficient Routing of Mobile Agents for Agent-Based Integrated Enterprise Management: A General Acceleration Technique," *Lecture Notes in Business Information Processing*, Vol. 88, 2011, pp. 1-20. doi:10.1007/978-3-642-24175-8_1

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," MIT Press, Cambridge, 2001.

[14] F. Ergun, R. Sinha and L. Zhang, "An Improved FPTAS for Restricted Shortest Path," *Information Processing Letters*, Vol. 83, No. 5, 2002, pp. 287-291. doi:10.1016/S0020-0190(02)00205-3

[15] R. Gibson and E. Schuyler, "Google Maps Hacks: Tips & Tools for Geographic Searching and Remixing (Hacks)," O'Reilly Media, Inc., 2006.

[16] A. Brooke, D. Kendrick and A. Meeraus, "GAMS: A User's Guide," The Scientific Press, Redwood City, 1998.

[17] S. Andrews, H. Wang, D. Ni, S. Gao and J. Collura, "Development and Implementation of an Adapted Evacuation Planning Methodology in the Framework of Emergency Management and Disaster Response: A Case Study Using TransCAD," *Journal of Transportation and Security*, Vol. 2, No. 4, 2010, pp. 352-368. doi:10.1080/19439962.2010.517743

[18] M. Pinedo, "Scheduling: Theory, Algorithms, and Systems," Prentice Hall, Englewood Cliffs, 1995.

[19] J. R. Evans, "An Efficient Implementation of the Wagner-Whitin Algorithm for Dynamic Lot-Sizing," *Journal of Operations Management*, Vol. 5, No. 2, 1985, pp. 229-235. doi:10.1016/0272-6963(85)90009-9

[20] A. D. Klose, "Facility Location Models for Distribution System Design," *European Journal of Operational Research*, Vol. 162, No. 1, 2005, pp. 4-29. doi:10.1016/j.ejor.2003.10.031

