# A Non-Deterministic Path Generation Algorithm for Traffic Networks

**Bo Zhou**

Doctoral Student
bzhou@gwu.edu

**Azim Eskandarian**

Director, Professor
eska@gwu.edu

Center for Intelligent Systems Research
The George Washington University
20101 Academic Way
Ashburn, VA 20147
Phone: (703) 726-8320
Fax: (703) 726-8505

## ABSTRACT

In many transportation applications it is useful to find multiple paths for an origin-destination pair. This paper presents a non-deterministic approach to generate alternative paths in traffic networks. The algorithm exhibits desirable features in both computational complexity and path quality. Hypothetic examples are provided to evaluate the generated paths in terms of diversity and efficiency. A microscopic traffic simulation is then used to introduce potential applications in transportation practices. Some future works are also discussed.

## INTRODUCTION

Finding the minimum-cost path from an origin to a destination in a network is a well-studied problem. Most existing solutions are based on Dijkstra's labeling method (Dijkstra 1959), adapting either the label-setting (LS) or the label-correcting (LC) procedure (Sheffi 1985). A slight variation to the basic Dijkstra's approach is the symmetrical or bi-directional Dijkstra algorithm (Cherkassky et al 1993). It performs a forward search from the origin and a backward search from the destination simultaneously, in an attempt to reduce the search complexity. When the underlying network is Euclidean, another search technique called the A* algorithm is often used (Sedgewick and Vitter 1986). The key idea is to integrate the inherent geometric information in order to bias a more directed search towards the destination. Although the A* algorithm on average improves the run time over the Dijkastra's algorithm, the solution is sub-optimum, i.e., it does not always find the minimum-cost path.

The minimum-cost path problems have significant applications in many fields of Intelligent Transportation Systems (ITS), especially in Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS) (Ziliaskopoulos and Mahmassani 1993). For example, an essential part of most in-vehicle Route Guidance Systems (RGS) currently under development is on-line calculation of the optimum route between a designated origin-destination (O-D) pair (Bekhor et al 2001). Herein the optimum route is usually the one with least expected travel time between the given O-D pair. The RGS consists of two sub-systems: the centralized system and the decentralized systems. The centralized system updates the link travel time regularly and calculates the optimum route in real-time upon receiving drivers' requests. The optimum route is then relayed to the decentralized systems for the drivers' use.

However, there are many situations that finding the minimum-cost path is not sufficient. Instead, it is necessary to identify some alternative paths for a given O-D pair. For example, currently most drivers are not equipped with the RGS and hence do not have complete information of the traffic network. Studies have shown that in such cases the route decisions are stochastic (Fu and Rilett 1998). In other words, the actual route taken by a particular driver is not necessarily the minimum-cost path. As another example, assume the perfect case where all drivers are guided by the RGS. In reality, some drivers may prefer one particular route to another for various reasons such as route familiarity, toll charges, etc. Therefore, it is more desirable that the RGS suggests several routes for each O-D pair and lets the driver make the choice. In addition, for transportation management purposes there are needs to take alternatives to the minimum-cost path. One particular example is that under emergency situations there are often excessive traffic demands from the affected areas to the safe areas. Part of the traffic should be diverted off

the minimum-cost route in order to reduce congestions and minimize the total evacuation time. Finally, finding alternative paths has important applications in transportation planning and analysis processes. In particular, stochastic traffic assignment models typically encompass a route generation phase, in which the set of efficient routes is generated for each O-D pair (Dial 1969). Then at the next phase traffic demand is assigned appropriately among those paths.

Current approaches for finding alternative paths fall into two main categories. The first is known as the K shortest paths method, where the goal is to identify the first, second, …, and Kth best path for a given O-D pair (Yen 1971). It was first proposed in the 1950' (Bellman 1958), and ever since then numerous algorithms have been developed as improvement. The biggest disadvantage of the K shortest paths method is its computational complexity (Yen 1971). It tends to run fairly slow, especially when large network is involved. This impedes its potential applications in ATIS where real-time route calculation is essential. Yet another drawback of it is that the generated paths tend to be quite similar, meaning they have a high percentage of shared links (Scott 1997). This is often undesirable because congestions may develop at the shared links if all traffic flow between the O-D pair is routed via the K shortest paths.

The second strategy for finding alternative paths involves a penalty concept (Scott 1997). Given an O-D pair, the algorithm first calculates the minimum-cost path using standard Dijkstra's approach. Then one or more links comprising the best path are chosen to be penalized by some factor $\lambda$ that is greater than one, meaning that the cost of each link is multiplied by $\lambda$. The newly calculated minimum-cost path for the updated network is regarded as an alternative to the best path. It has been shown that the penalty method is able to run as fast as the standard Dijkstra's algorithm, and it tends to avoid the similarity problem that the K shortest path method suffers (Scott 1997). As an extreme variation of the penalty method is the link elimination method (Azevedo et al 1993). In this case every chosen link is removed from the network rather than increasing its cost. Both the penalty method and the elimination method have disadvantages. First the algorithm must explicitly specify which particular links should be chosen for penalization or elimination. This choice usually affects the quality of the generated path. In the case of link elimination, it is even possible that deleting some links may cause the network to lose the necessary connectivity for having any path between the O-D pair. Another issue is that only the links comprising the initial best path are modified while the rest of the whole network remains unaffected. Therefore, the method tends to find only a very limited number of alternative paths. In other words, if the method is applied many times in an attempt to generate different paths, presumably it will keep finding a small set of paths over and over again.

In this paper, a new technique is proposed for generating alternative paths in traffic networks. The new approach is designed to achieve two main objectives. The first is that it must be computationally efficient so as to be applicable for large real traffic networks. The second goal is that the generated paths should be diverse and efficient. The paths are considered diverse if they do not overlap with one another significantly. The paths are considered efficient if none is associated with an unacceptable high cost. Both criterion are carefully defined and evaluated in this study.

The new algorithm finds an alternative path in traffic network at three stages. At the first stage, the program internally determines the new locations for the designated O-D pair. Specifically, one of the outbound links from the origin node is randomly chosen and the downstream node of the link is set to be the new origin. Similarly, one of the inbound links to the destination node is randomly chosen and the upstream node of the link is set to be the new destination. This treatment is included to fulfill path diversity requirement. At the next stage, the program uses a Dijkstra's type labeling method for searching a path between the new O-D pair. As a result, the complexity is equal to the Dijkstra's algorithm. In order to get a different path than the minimum-cost one, the program assigns a random factor to the cost of each link as it is scanned by the labeling algorithm. The use of randomized link cost is shown to meet both the path diversity and efficiency requirements. Finally, the path from the initial origin to the new origin, the path from the new origin to the new destination, and the path from the new destination to the initial destination are combined to form an alternative path.

The rest of this paper is organized as follows. Section 2 presents the detailed description of the new algorithm. Section 3 evaluates the quality of the generated paths in two hypothetic cases. A microscopic traffic simulation of a nuclear power plant evacuation-planning scenario is given in Section 4 to demonstrate the possible applications of the new algorithm in real world. Finally conclusions and future works are discussed in Section 5.

## PATH GENERATION

New algorithms for finding alternative paths in traffic networks are needed because existing tools either are computationally too expensive, or fail to ensure the path diversity. We propose a new algorithm that addresses both issues.

### Problem Definition

As depicted in figure 1, a traffic network is represented as a directed graph *G(N,A)*, where *N* is a finite set of nodes and *A* is a finite set of links. A link is the connection between two different nodes. The starting node is called the upstream node and the end node is called the downstream node. Each link is associated with a cost, which could be distance, time, toll, etc. In traffic networks, the most relevant cost is the travel time and hence is used here. Depending on applications, the travel time could also include the projected turning delay at intersections or other constraint. A path between two nodes is defined as a sequence of nodes representing the sequence of links that connects from the origin to the destination. The cost of the path is simply the summation of all the link costs. The graph *G* is assumed to be connected, i.e., there exists at least one path between any two nodes (Diestel 2000). This is a reasonable assumption as it is the case in most traffic networks.

INSERT FIGURE 1 NEAR HERE

An origin node *s* depicted a square box and a destination node *t* depicted a triangle are designated, as shown in figure 1. All other nodes are depicted a circle with the node number inside it. Each line represents a two-way link, and the number above the line is the cost for both links. In this case, the minimum-cost path between *s* and *t* is *s-1-2-3-4-8-9-t*, denoted *z*. We want to find an arbitrary path other than *z* between the O-D pair. A path is acceptable only if it is efficient. For example, the path *s-1-2-6-8-4-5-7-9-t* is not efficient because it takes an undesirable detour away the destination. Moreover, the generated path should not have too many shared links with z. In other words, running the algorithm multiple times should give different paths.

### A Three-Stage Algorithm

The network is represented in the forward star form (Sheffi 1985). Given an O-D pair, our new algorithm finds an alternative path in three stages:

1. Setting new origin and destination;
2. Finding a path between the new O-D pair;
3. Completing the path by combination.

The algorithm is non-deterministic such that two executions usually result in two different paths. Therefore, it could be effectively used to generate a set of alternative paths for an O-D pair.

**Setting New Origin and Destination** For any path between an origin s and a destination t, one fact is certain: it always starts at s and terminates at t. Suppose that $D = \{d_1, d_2, \cdots, d_m\}$ is the set of immediate downstream nodes of s, and $U = \{u_1, u_2, \cdots, u_n\}$ is the set of immediate upstream nodes of t. Then an arbitrary path p between s and t can be represented as $s - d_i - \cdots - u_j - t$, where $1 \le i \le m$ and $1 \le j \le n$. In order to maximize the diversity of p, it is obvious that i and j should be as random as possible. However, if the link costs are severely biased around s or t the searching algorithm is very likely to choose one particular node in D or U while ignoring the others. For example, in figure 1 D = {1, 3, 5, 6} for node 2. However, the cost of link 2-3 is only 1, while the costs of 2-1, 2-5 and 2-6 are 11, 10 and 15 respectively. In this case, any searching algorithm based on Dijkstra's labeling method will favor node 3 while nodes 1, 5 and 6 are most likely abandoned. To address this issue, our algorithm forces the program to select the nodes in D and U on an equal basis. This is done by randomly pick the number i and j, and then use $d_i$ and $u_j$ as the new O-D pair.

In practice, the origin *s* may have only one downstream node $s_1$, or $m = 1$. Under such situation, the program will first move the new origin to $s_1$, and then temporarily delete the link of $s_1$-*s*, if it exists. Next it will test if $s_1$ has only one downstream node. If so it will just move the new origin to the only downstream node of $s_1$,

denoted $s_2$. Otherwise, it will identify the set $D$ for $s_1$, and randomly pick one node $d_i$ in $D$ as the new origin. Again it will temporarily delete the link of $d_i$-$s_1$, if it exists. The logic for link deletion as moving the origin to the next location is straightforward. The deletion ensures that the program do not search backwards to the previous location. Similarly, if the destination t has only one upstream node $u_1$, then the new destination is moved to $u_1$ and the link $u$-$u_1$ is deleted temporarily. The process repeats until the set $U$ of the new destination has at least two nodes. Then one node is randomly chosen as the new destination, and the link from the previous destination to the new destination is temporarily removed.

To illustrate the whole process, again take the case in figure 1 as an example. First the origin is moved from *s* to node 1 since it is the only downstream node of *s*. Then the link *1-s* is deleted, which makes node 2 the only downstream node of node 1. Hence the origin is further moved to node 2 and then the link *2-1* is deleted. Now node 2 has three downstream nodes or $U = \{3, 5, 6\}$. The program then randomly picks one node from $U$, say node 6, and deletes the link *6-2*. So finally the new origin is node 6.

**Finding a Path between the New O-D Pair** To find a path between the new origin and destination, a Dijkstra's type label-correcting method is used because of its efficiency (Sheffi 1985). The label-correcting method finds the shortest path from the origin node to all other nodes in the network. It essentially maintains a minimum-path tree with the origin node being the root, and continuously updates the tree by iteratively scanning the network nodes (Sheffi 1985). As each node is scanned, the corresponding link cost is used to determine if a better path up to that node exists. The algorithm terminates as soon as no better improvement can be made to the minimum-path tree.

However, since the interest is to find a path not necessarily having the minimum cost, a slight modification to the standard label-correcting procedure is made. As a link is being scanned, a random factor λ is assigned to that link. A new quantity that equals to the corresponding link cost multiplied by λ is then used to update the minimum-path tree. Note the link cost is actually kept unchanged. Apparently the random factor λ has a significant effect on the resulting path. In our study, λ is set to be a natural number which is in the range of one to some upper bound δ. Experiments are designed to investigate how the value of δ affects the quality of path, as will next be shown in Section 3.

INSERT FIGURE 2 NEAR HERE

**Completing the Path by Combination** The final step is straightforward. The path from the initial origin to the new origin, the calculated path from the new origin to the new destination, and the path from the new destination to the initial destination are connected to form a complete path. Before exiting the program, all the deleted links at stage one are restored. This ensures that the initial network connectivity is always correctly represented at the beginning of each run.

The complete algorithm is summarized as pseudo code in figure 2. The details of the label-correcting procedure are not shown since it is a well-established approach (Sheffi 1985). The algorithm is implemented in C++ due to its run-time efficiency.

## PATH EVALUATION

Unlike many existing path-finding methods, the new algorithm is non-deterministic. For any given O-D pair, running the algorithm twice usually gives two different paths. To evaluate the performance of the algorithm, we first define the criterion to assess the quality of individual path. The average quality of paths under various situations is then studied.

### Path Evaluation Criterion

Let *z* denote the minimum-cost path between a given O-D pair, and let *p* denote a path generated by the algorithm. The path *p* is considered efficient if it has a relatively low cost and is different than z. The first criteria denoted the cost ratio is the ratio between the cost of *p* and the cost of *z*

$$cost\_ratio(p) = cost(p)/cost(z)$$

It is obvious that the cost ratio of any path is at least one. A higher cost ratio suggests that *p* has much longer length than *z* and is thus less desirable as an alternative path.

The second criteria denoted the shared ratio is expressed as

$$share\_ratio(p) = num\_links(p,z)/num\_links(z)$$

The numerator is the number of links that *p* and *z* both have, and the denominator is the number of links in *z*. Apparently the share ratio is at most one. A higher share ratio means that *p* has more shared links with *z*, and is hence a less favorable substitute to *z*.

Both criteria are used to measure individual path quality. Because the algorithm is non-deterministic, the average values of the criteria are more relevant to evaluate the solutions to a given problem.

**Factors that Affect the Path Quality**

There are several factors that may affect the generated path quality. Careful study of these factors is crucial for evaluating the performance of the new algorithm.

**The Random Factor** As described in section 2, the new algorithm assigns a random number λ to each link being scanned, where λ is in the range of one to an upper limit δ. A greater value of δ means that the link cost is more randomized, therefore the resulting path tend to be less similar than the minimum-cost one. In other words, the algorithm will be able to explore a larger part of the network and hence more different paths are likely to be found. However, as the path become less similar than the minimum-cost one, the cost ratio tends to rise which is not very desirable. Therefore, there might be an optimum value of δ that maximizes the path quality on average. In our experiments, δ is set to different values to test if such hypothesis is true.

**Network Structures** The structure of traffic networks is another factor to consider when evaluating the general performance of a path-finding algorithm. Although the actual topologies of traffic networks are virtually infinite, they could be distinguished into three main patterns as the grid-like, the star-like and irregular patterns (Jiang and Claramunt 2004). In the grid-like pattern, illustrated in figure 3, links are either parallel or perpendicular to one another. In the star-like pattern, illustrated in figure 4, there are some dense points at which several links converge. The irregular pattern is merely a rough mixture of the first two. In this study, both the grid-like and the star-like patterns are used as the underlying networks.

INSERT FIGURE 3 & 4 NEAR HERE

**Experiments and Results**

Both the grid-like network and the star-like network are used as our test networks, as shown in figure 3 and figure 4 respectively. In both figures, the origin node is depicted a square box and the destination node is depicted a triangle. All other nodes are depicted as a circle. Each line represents a two-way link. The link cost is not shown for simplicity, but it is assumed to be proportional to the geographical distance.

For each test network, several experiments are done. The algorithm is repeated 10, 100 and 1000 times respectively. In addition, the value of δ is varied from 2 to 10. Several measurements are collected. The number of unique paths is compared with the number of executions to show the effectiveness of finding alternatives. The distribution of both the cost ratio and the share ratio are also recorded as indicators of the average path quality.

Table 1 summarizes the collective results from the experiments. Firstly it is shown that the algorithm finds different paths successfully. With only 10 runs, the unique paths generated in both networks are close to 10 under all cases. As the number of total runs increases the number of unique paths also increases. Secondly it is shown that the quality of generated paths is very satisfactory. In all cases, most paths have a cost-ratio less than 1.5, with only a few exceptions that are close to 2. On the other hand, the share-ratio is as low as 20 percent, suggesting many paths are highly different from the minimum-cost path. Thirdly it is seen that the value of δ does have a noticeable effect on the solutions. Generally as δ increases, more unique paths are found at the expense of higher cost-ratio. However,

even when δ is 10, the resulting paths are still quite efficient as the maximum cost-ratio is still less than 2 in the worst case. In practice the value of δ could be controlled in order to achieve optimum results in both path efficiency and diversity. Finally, the results show that the two networks have very different properties. The paths found in the grid-like network are more similar as the minimum-cost path, while those in the star-like network are very diverse with a much wider range of cost-ratio and share-ratio. This is clearly due to the different structure of the networks.

INSERT FIGURE 5 & 6 NEAR HERE

Figure 5 and 6 present the distributions of the path criterion for each network when the total number of runs is 100, and the value of δ is selected to be 2 and 5. In the grid-like network as shown in Figure 5, it is clearly seen that the majority of paths have a very low cost-ratio. When the value of δ is 2, the cost ratio is very close to 1. When the value of δ is 5, the cost ratio increases in general but is well below 1.2, which is very desirable. In both cases the share-ratio demonstrate a very random distribution between a wide ranges. The share ratio of most paths is between 20% and 60%, with only a few close to 80%. This shows that the paths are highly diverse. In the star-network as shown in Figure 6, the behavior is much more random. Nevertheless, the cost-ratio is still well below 2 with most between 1.2 and 1.6. The share ratio is even more desirable as most are below 60%.

Overall, the experimental results suggest that the algorithm is efficient to find alternative paths for different network structures. The average path quality is very satisfactory as both the average cost ratio and share ratio are low. Therefore, the algorithm can be used to generate one or more paths for various applications.

## A DEMONSTRATION PROBLEM

Traffic models are important tools for emergency evacuation planning and analysis (Franzese and Han 2002). The route choice module is a critical component of most traffic models. It essentially calculates different routes and assigns them to drivers in order to achieve optimum simulation results. As an example, a preliminary microscopic traffic model is developed to show the usefulness of alternative paths for regional evacuation simulations.

### A Realistic Evacuation Scenario

Figure 7 illustrates a realistic regional evacuation scenario near the Twin cities, Minnesota (Lu and Shekhar 2002). The Monticello nuclear power plant is located in a populated area. Effective evacuation plans are highly crucial to ensure the safety of nearby populations if any nuclear accident happens. In the figure, the location of the power plant is indicted with a red square box. The affected cities subject to evacuation are indicated with orange circles. The designated shelter place for all the evacuees is indicated with a green triangle, which is about 40 miles away from the power plant.

The assumed number of vehicular evacuees from each origin city is listed in table 2. The cities are numbered as shown in figure 7.

INSERT FIGURE 7 NEAR HERE

### The Traffic Model

The TSIS Corsim is a validated microscopic traffic simulator which is widely used to support transportation design and analysis (ITT Industries 2003). Corsim is able to capture the detailed individual vehicular movements at every simulation interval. It also generates aggregated results as measures of effectiveness (MOE) at the end of each simulation period.

When used for evacuation modeling, the traffic model could adopt either static traffic assignment (STA) or dynamic traffic assignment (DTA) as its route choice component (Hobeika and Kim 1998). In a STA model, the traffic conditions at the beginning of the simulation are assumed to be unchanged throughout the entire simulation period. This is obviously an invalid assumption and hence STA is unable to reflect the dynamics of traffic. The DTA, on the other hand, uses continuously updated network information to make the route choice. Therefore, it is a better representation of the traffic flow.

In this demonstration problem, we develop a very preliminary Corsim model that uses neither STA nor DTA. Instead, the evacuees are assigned to a random path generated by our new algorithm at each loading interval. The random assignment is used in an attempt to mimic the stochastic route choice in reality. It also tends to reduce congestions on the heavily used roads.

**Simulation Results**

In the Corsim model, all roads are assumed to be 2-lane, with a free flow speed of 60 mph. The total loading period is assumed to be 30 minutes, and is divided equally into 15 loading intervals. Constant loading rate is assumed. At each loading interval, a group of vehicles from each origin is loaded onto the network. A new path created by our new algorithm is assigned to the emerging traffic. The vehicle loading and route assignment is done by the Corsim path-following capabilities (TSIS User Support Website 2003).

Another simulation, which uses the Corsim static traffic assignment, is used for comparison purposes. The simulation is divided into two periods. In the first period the identical traffic demand as in the first case is loaded by specifying the corresponding O-D table. In the second period no more traffic is loaded and the simulation continues until the network is cleared.

The simulations are performed on a desktop computer with a 2.4 GHz Intel Xeon processor and 1024 MB of system memory. The MOE for both simulations are summarized in table 2. The network clearance time is the total time from the start of simulation till all evacuees arrive at the destination. The average speed is the mean travel speed of all vehicles. The average delay percentage is the total vehicular delay time divided by the total vehicular travel time. The maximum queue length is the maximum number of vehicles queued on some link during simulation. The comparison shows that overall the simulation with generated paths gives a better result. The main reason is that the Corsim static traffic assignment is done at the beginning of the simulation, and only accounts for the free-flow network conditions. Therefore, the assignment directs a large portion of traffic through the initial shortest path of each O-D pair. The generated paths simulation, however, spreads the traffic over a set of different paths and thus effectively reduces congestions on certain links.

## CONCLUSIONS AND FUTURE WORKS

In this paper, a new algorithm for finding alternative paths in traffic networks was presented. The method modifies a Dijkstra's type label-correcting procedure by introducing a random factor for each link being scanned. Two criteria were used as the primary indicator of the generated path acceptability. The algorithm was evaluated using two characteristic network patterns. A realistic evacuation scenario was used to demonstrate the effectiveness of the new algorithm.

The new algorithm is readily scalable to more sophisticated applications. For example, intersection turning delays can be added to the link cost function to better represent the travel time. As another example, if redundant paths are not allowed, an extra path-checking procedure following the algorithm would well solve the problem. If a certain node or link must be included in any path, a straightforward solution is dividing the target path into two parts at the desired node or link and using the algorithm to find a path for each part before connecting them together. On contrary, if a certain node or link must be excluded from any path, one solution is first using the algorithm to generate a set of initial candidate paths, and then eliminating those which include the node or link.

There are a few potential improvements over the current approach. As has been shown in section 3, the generated paths for different network structures exhibit highly different natures. Currently the users can manually adjust the random factor to improve the path quality. It would be nice if the algorithm is able to detect the network structure and automatically determine the optimum random factor. Another issue is associated with the run time speed. When the underlying network is extremely large, the label-correcting method tends to run slow. In this case, the label-correcting part could be replaced by the A* method as the searching engine.

## BIBLIOGRAPHY

Azevedo, J., Costa, M. S., Madeira, J. S. and Martins, E. V., "An Algorithm for the Ranking of Shortest Paths," *European Journal of Operational Research* 69: 97–106, 1993.

Bekhor, S., Ben-Akiva, M. and Ramming, M., "Estimating Route Choice Models for Large Urban Networks," 9th World Conference on Transport Research, Seoul, Korea, 2001.

Bellman, R. E., "On a routing problem," *Quart. Appl. Math*, 16:87–90, 1958.

Cherkassky, B. V., Goldberg, A.V. and Radzik, T., "Shortest Paths Algorithms: Theory and Experimental Evaluation," *Technical Report 93-1480*, Computer Science Department, Stanford University, 1993.

Dial, R., "Algorithm 360: Shortest Path Forest with Topological Ordering," *Communications of ACM*, Vol. 12, pp. 632-633, 1969.

Diestel, R., "Graph Theory," volume 173 of Graduate Texts in Mathematics, Springer Verlag, 2nd Edition, 2000.
Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik*, Vol. 1, pp. 269-271, 1959.

Franzese, O. and Han, L., "Using Traffic Simulation for Emergency and Disaster Evacuation Planning," 81st Annual Meeting of the Transportation Research Board, Washington, D.C., January 2002.

Fu, L., Rilett, L. R., "Expected Shortest Paths in Dynamic and Stochastic Traffic Networks," *Transportation Research - B* 32(7), 499-516, 1998.

Hobeika, A.G. and Kim, C., "Comparison of Traffic Assignments in Evacuation Modeling," *IEEE Transactions on Engineering Management*, 45(2): 192-198, 1998.

ITT Industries, Inc., Systems Division, "TSIS 5.1 User's Guide," Colorado Springs, C.O., 2003.

Jiang, B. and Claramunt, C., "A Structural Approach to the Model Generalization of an Urban Street Network," *GeoInformatic* 8(2): 157-171, 2004.

Lu, Q.S. and Shekhar, S., "Capacity Constrained Routing for Evacuation Planning: New Results," Intelligent Transportation Systems Safety and Security Conference, Miami, Florida, March 24-25, 2004.

Scott, K., Pabón-Jiménez, G. and Bernstein, D., "Finding Alternatives to the Best Path," 76th Annual Meeting of the Transportation Research Board, Washington, D.C., 1997.

Sedgewick, R. and Vitter, J., "Shortest Paths in Euclidean Graphs," *Algorithmica*, Vol. 1, No. 1, pp. 31-48, 1986.

Sheffi, Y., "Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods," Prentice Hall, Englewood Cliffs, NJ, 1985.

TSIS User Support Website, "The Path Following Capability in Corsim," www.fhwa-tsis.com/path%20following%20capability%20in%20CORSIM%205_2.htm, Accessed June 2003.

Yen, J.Y., "Finding the K Shortest Loopless Paths in a Network," *Management Science*, Vol. 17, No. 11, pp. 712-716, 1971.

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

List of Figures

List of Tables

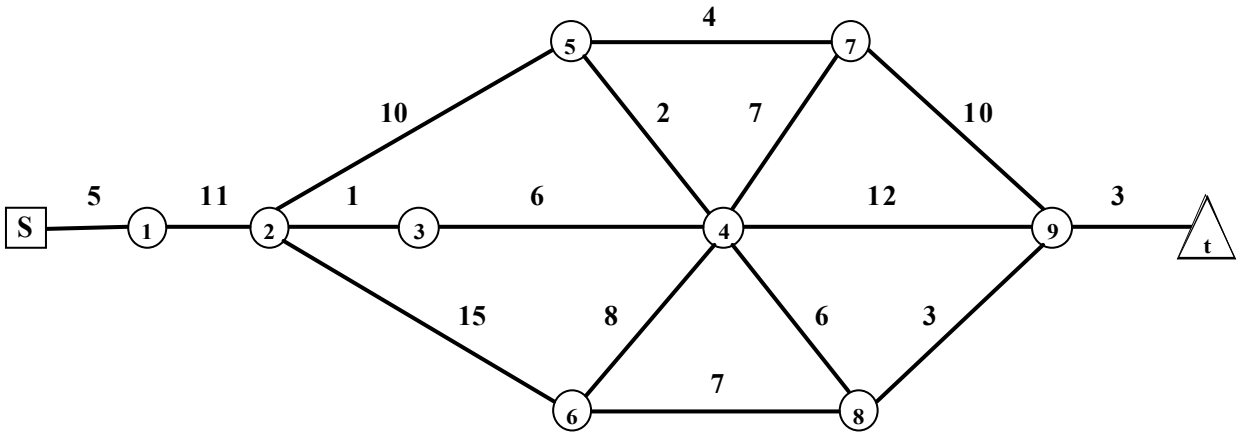


**Figure 1. Representation of an Example Traffic Network**

```
Input: network G, origin s and destination t;
Output: an alternative path from s to t;
begin
  if origin has only one downstream node, repeat
    let s' be the only downstream node;
    set s = s';
    delete link s'-s if it exists;
  endif
  let D be the set of all downstream nodes of the origin;
  randomly choose one node d from D, and set new origin = d;
  delete link d-s if it exists;
  if destination has only one upstream node, repeat
    let t' be the only upstream node;
    set t = t';
    delete link t-t' if it exists;
  endif
  let U be the set of all upstream nodes of the destination;
  randomly choose one node u from U, and set new destination = u;
  delete link t-u if it exists;
  begin finding a path between d and u
    for each link being scanned
      multiply its cost with a random number λ
  end finding a path between d and u
  final path = path (s to d) + path (d to u) + path (u to t);
  restore all deleted links;
end
```

**Figure 2. Pseudo-code of the Algorithm**

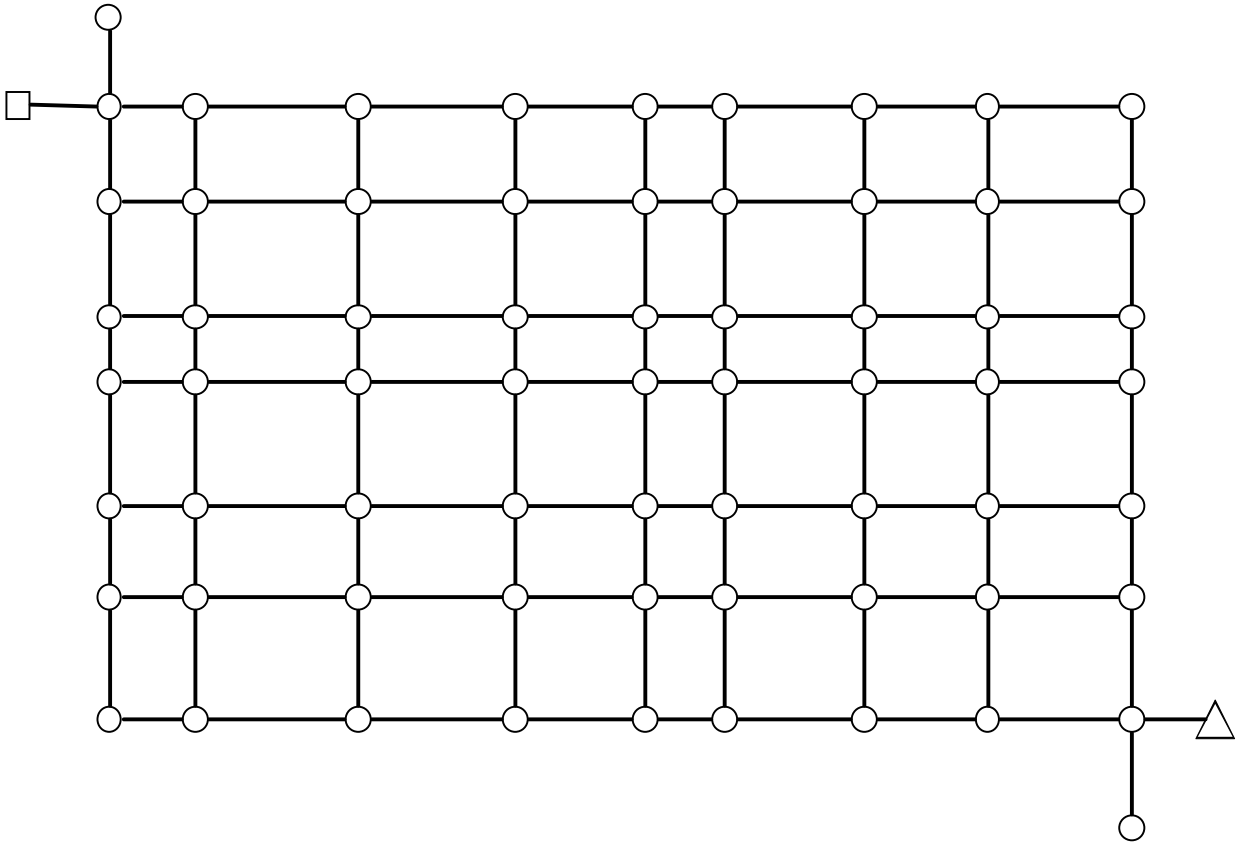

**Figure 3. An Example of Grid-like Traffic Networks**

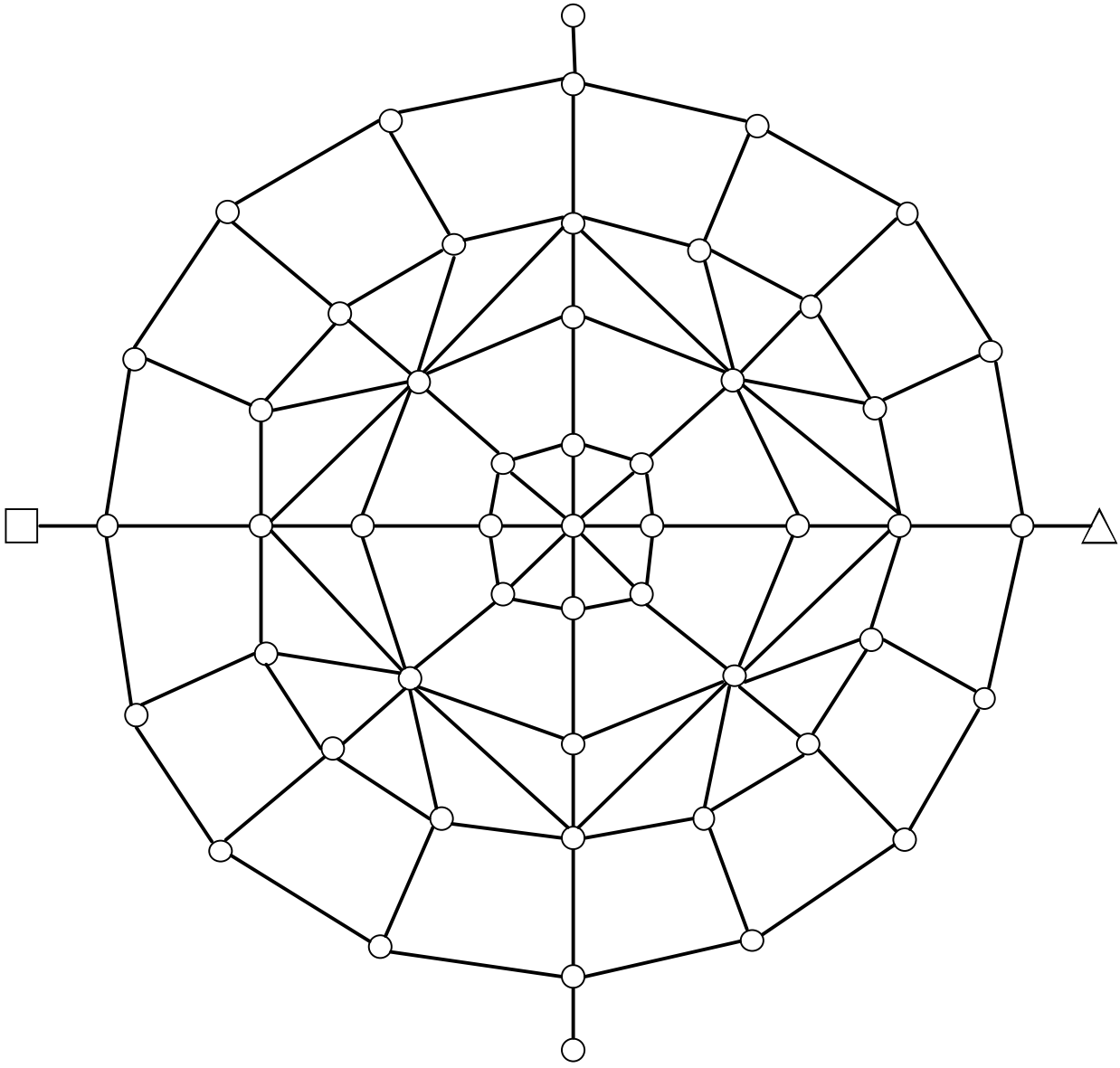

**Figure 4. An Example of Star-like Traffic Networks**

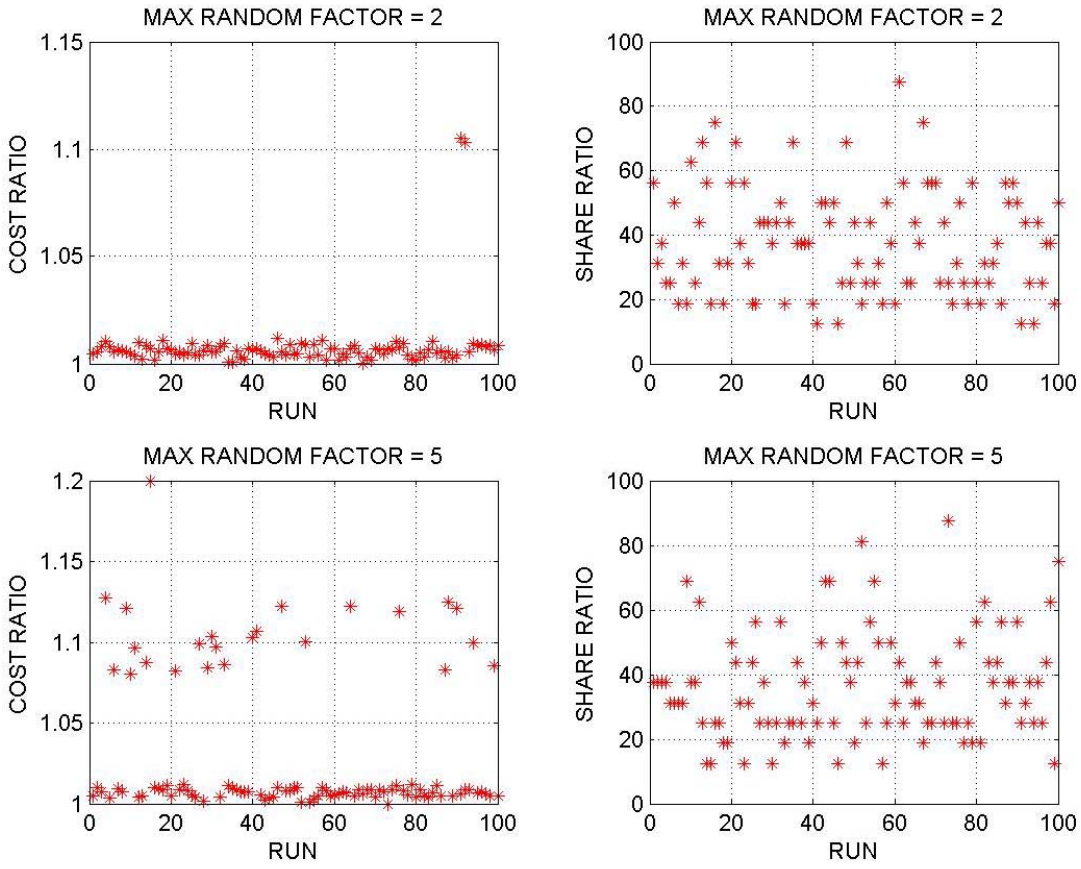


**Figure 5. Distribution of Path Criterion in Grid-like Network**

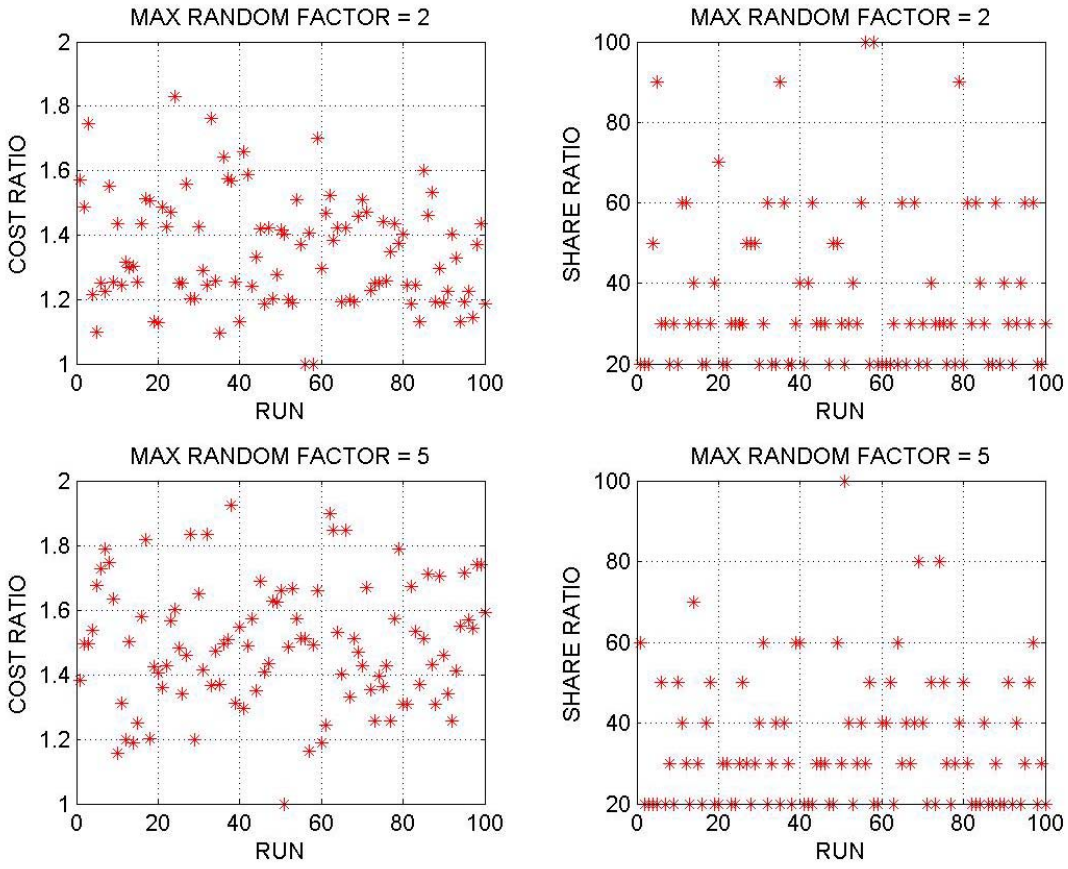


**Figure 6. Distribution of Path Criterion in Star-like Network**

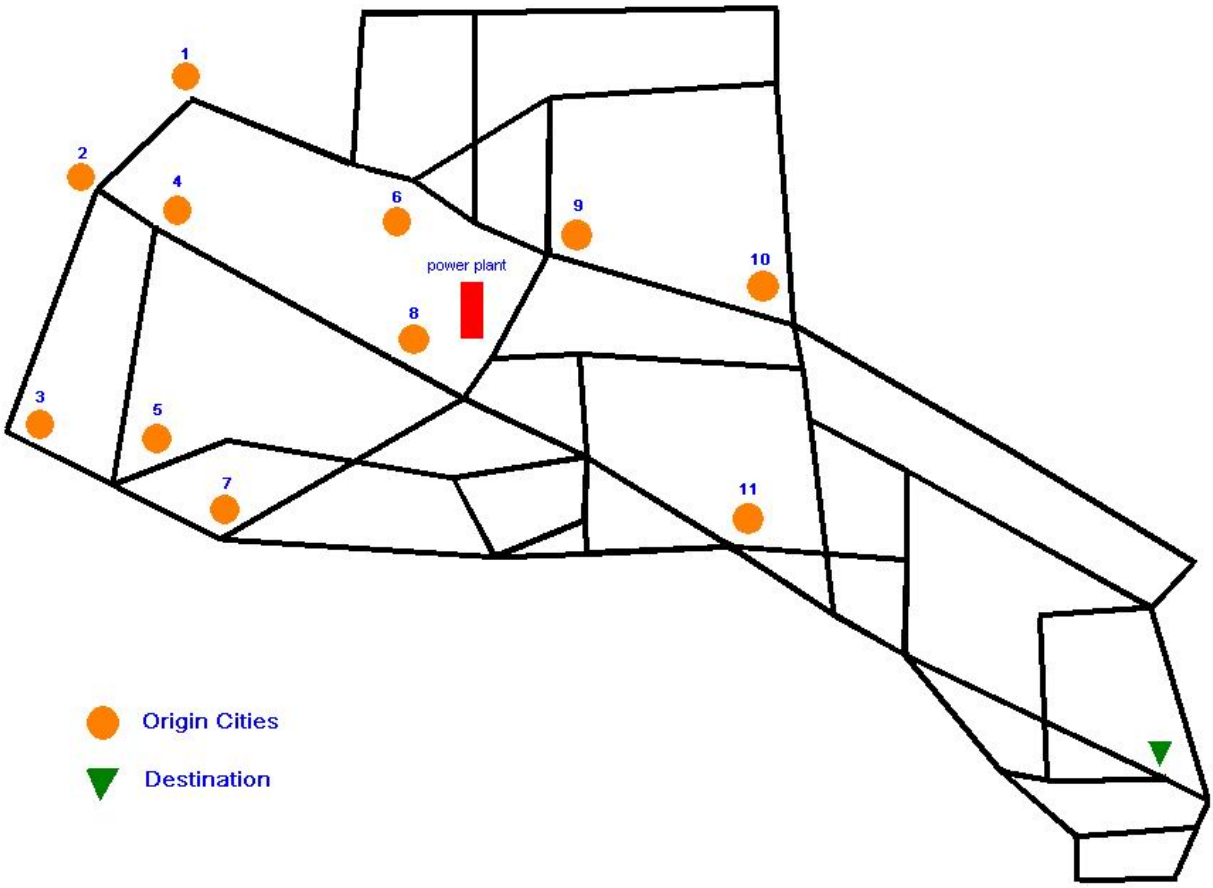


**Figure 7. Network Representation of the Evacuation Scenario**

**Table 1 Collective Measurements for Generated Paths**

| Grid-like Network | | | | | | |
|---|---|---|---|---|---|---|
| **number of runs** | **δ** | **number of unique paths** | **cost ratio** | | **share ratio (%)** | |
| | | | **min** | **max** | **min** | **max** |
| 10 | 2 | 10 | 1.0032 | 1.0829 | 12.50 | 62.50 |
| | 5 | 9 | 1.0032 | 1.1178 | 18.75 | 50.00 |
| | 10 | 10 | 1.0025 | 1.0973 | 12.50 | 68.75 |
| 100 | 2 | 46 | 1.0001 | 1.1050 | 12.50 | 87.50 |
| | 5 | 71 | 1.0000 | 1.1999 | 12.50 | 87.50 |
| | 10 | 62 | 1.0001 | 1.1957 | 12.50 | 75.00 |
| 1000 | 2 | 105 | 1.0001 | 1.2006 | 12.50 | 87.50 |
| | 5 | 216 | 1.0000 | 1.2848 | 12.50 | 87.50 |
| | 10 | 262 | 1.0008 | 1.3073 | 12.50 | 87.50 |
| Star-like Network | | | | | | |
| **number of runs** | **δ** | **number of unique paths** | **cost ratio** | | **share ratio (%)** | |
| | | | **min** | **max** | **min** | **max** |
| 10 | 2 | 8 | 1.2258 | 1.6970 | 20.00 | 60.00 |
| | 5 | 10 | 1.2164 | 2.0264 | 20.00 | 60.00 |
| | 10 | 10 | 1.2027 | 1.9191 | 20.00 | 60.00 |
| 100 | 2 | 68 | 1.0000 | 1.8290 | 20.00 | 100.00 |
| | 5 | 94 | 1.0000 | 1.9253 | 20.00 | 100.00 |
| | 10 | 96 | 1.0000 | 2.1407 | 20.00 | 100.00 |
| 1000 | 2 | 307 | 1.0000 | 1.8028 | 20.00 | 100.00 |
| | 5 | 572 | 1.0000 | 2.1298 | 20.00 | 100.00 |
| | 10 | 636 | 1.0000 | 2.1521 | 20.00 | 100.00 |

**Table 2 Selected Simulation Results for the Evacuation Problem**

| Number of vehicular evacuees at each source city | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| 450 | 750 | 300 | 1050 | 300 | 600 | 450 | 1500 | 1200 | 300 | 750 |
| Comparisons of MOE | | | | | | | | | | |
| | | | | **Path-following simulation** | | | **Traffic assignment simulation** | | | |
| **Network clearance time** | | | | 3 hours 10 minutes | | | 3 hours 45 minutes | | | |
| **Average speed (mph)** | | | | 29.42 | | | 28.36 | | | |
| **Average delay percentage (%)** | | | | 0.51 | | | 0.53 | | | |
| **Maximum queue length (vehicle)** | | | | 474 | | | 500 | | | |