Least Expected Time Paths in Stochastic, Time-Varying Transportation Networks

ELISE D. MILLER-HOOKS

Department of Civil and Environmental Engineering, The Pennsylvania State University, University Park, Pennsylvania 16802

HANI S. MAHMASSANI

The University of Texas at Austin, Austin, Texas 78712

We consider stochastic, time-varying transportation networks, where the arc weights (arc travel times) are random variables with probability distribution functions that vary with time. Efficient procedures are widely available for determining least time paths in deterministic networks. In stochastic but time-invariant networks, least expected time paths can be determined by setting each random arc weight to its expected value and solving an equivalent deterministic problem. This paper addresses the problem of determining least expected time paths in stochastic, time-varying networks. Two procedures are presented. The first procedure determines the a priori least expected time paths from all origins to a single destination for each departure time in the peak period. The second procedure determines lower bounds on the expected times of these a priori least expected time paths. This procedure determines an exact solution for the problem where the driver is permitted to react to revealed travel times on traveled links en route, i.e., in a time-adaptive route choice framework. Modifications to each of these procedures for determining least expected cost (where cost is not necessarily travel time) paths and lower bounds on the expected costs of these paths are given. Extensive numerical tests are conducted to illustrate the algorithms' computational performance as well as the properties of the solution.

The routing of critical service vehicles (EMS, police, fire) and hazardous shipments must often consider stochasticity of traffic conditions and other risks in transportation networks. Two-way communication coupled with ITS technologies often allow vehicles to select paths dynamically, as congestion unfolds. In situations where future travel times are at best known a priori with uncertainty and the level of congestion induces systematic time dependence of the travel time distributions, the travel times on the network should be represented as random variables with probability distribution functions that vary with time. The resulting stochastic, time-varying (time-dependent) (STV) network will provide a more appropriate representation of actual conditions on which to base critical routing decisions than commonly used deterministic, static models.

Unlike deterministic networks, in which one can determine a single minimum time path between an origin and a destination, several paths may each have some positive probability of having the least time for some realization of the network when the arc times are stochastic. Thus, a set of Pareto-optimal (otherwise referred to as nondominated, or efficient) paths can be identified. For some applications, especially those of a repetitive nature, it may be sufficient to determine the paths with the least expected time (LET). The determination of LET paths in STV networks is more difficult than in networks where the arc travel time distributions are time invariant. In the latter, the LET path can be determined by setting each random arc weight to its expected value and solving an equivalent deterministic problem. One cannot simply set each arc weight random variable to its expected value at each time interval and solve an equivalent time-dependent shortest path problem (HALL, 1986). This paper addresses the problem of determining the LET paths in such STV networks.

Transportation Science, © 2000 INFORMS
Vol. 34, No. 2, May 2000 2000

0041-1655/00/3402-0198 $05.00
pp. 198–215, 1526-5447 electronic ISSN

Several papers have addressed the problem of determining shortest paths in stochastic, stationary networks. FRANK (1969) derived a closed form solution for the probability distribution function of the minimum path travel time through a stochastic, time-invariant network. A number of other works address similar problems (SIGAL, PRITSKER, and SOLBERG, 1980; KULKARNI, 1986; COREA and KULKARNI, 1993; and others). LOUI (1983), EIGER, MIRCHANDANI, and SOROUSH (1985), MIRCHANDANI and SOROUSH (1985), and MURTHY and SARKAR (1996) present procedures for determining optimal paths in stochastic, time-invariant networks where the decision-maker's preferences are represented by utility functions of various forms. In addition, a large number of works have addressed Program Evaluation and Review Technique (PERT) networks in conjunction with project planning activities (VAN SLYKE, 1963; MALCOLM, ROSEBOOM, and CLARK, 1959; CHARNES, COOPER, and THOMPSON, 1964).

For STV networks, Hall (1986) proposed an approach combining branch-and-bound and K-shortest path techniques for determining the a priori LET path between an origin and a destination. The algorithm requires that the expected times and least possible times be calculated for each path; however, no procedure is given for calculating these values. The procedure requires calculation of the minimum possible time of every path, and does not guarantee that it will terminate before the expected value of all paths have been evaluated. The procedure does not consider the possibility that the LET path could have one or more cycles; therefore, it applies only to acyclic networks or cyclic networks with first in first out (FIFO)[1] arc weights. In the same work, Hall proposed a dynamic programming approach for determining optimal strategies for en-route decisions in response to experienced travel times on traveled arcs. He recognized that such time-adaptive route choice could follow paths with lower expected travel times than a priori optimization. This is identical to a recourse problem where recourse decisions can be taken once the value of one or more random variables are realized (see BIRGE and LOUVEAUX (1997) on stochastic programming). For a similar notion of optimality, PSARAFTIS and TSITSIKLIS (1993) considered optimal policies for specifying the least expected cost path between an origin and a destination in acyclic, dynamic, and stochastic networks. The cost of traveling on arcs leaving each node is a function of the state of that node, which, known only upon arrival at that node, varies randomly but independently of the states of the other network nodes.

MILLER-HOOKS and MAHMASSANI (1998) presented two efficient procedures for determining the least possible time paths in STV networks. Such paths may not necessarily be the most desirable in all applications because they do not consider all relevant risk dimensions. A procedure for determining paths with wider applicability, such as LET paths, is desirable.

In this paper, two specialized modified label-correcting algorithms[2] are presented for the problem of generating LET paths in STV networks. First, the expected value (EV) algorithm, is presented for generating all a priori LET paths with their associated expected times from all origins to a single destination for each departure time in a given period. Although the worst-case computational complexity of this algorithm is nonpolynomial, the average performance for networks with an average in- and outdegree each of four (as in common street networks) is shown experimentally to be considerably better than predicted under worst-case complexity analysis. In addition, cycles and non-FIFO arcs are permitted. Second, the expected lower bound (ELB) algorithm, is an efficient procedure for determining lower bounds on the expected times of the LET paths from all origins to a single destination for each departure time in the period, but without any associated path information. Minor additional information can be retained to enable the ELB algorithm to generate an exact solution for the problem of determining LET paths in a time-adaptive route choice framework. In this context, for a given origin–destination pair, at a specific departure time, a single path may not provide an adequate solution. This is because the optimal path depends on intermediate information concerning realized travel times on traveled arcs. Thus, a set of strategies, represented by acyclic subnetworks, referred to as hyperpaths (NGUYEN and PALLOTTINO, 1986), are generated to provide directions to the destination node conditioned upon arrival times at intermediate locations. Additionally, modifications to the EV and ELB algorithms are given for determining least expected cost (other than travel time) paths with their expected

[1]A definition of FIFO arc weights in stochastic, time-dependent networks is given in Miller-Hooks and Mahmassani (1998) by the following extension to the consistency assumption of KAUFMAN and SMITH (1993) in time-varying, deterministic networks: For any arc $(i, j) \in \mathcal{A}$, $\Pr\{s + \tau_{ij}(s) \leq t + \tau_{ij}(t)\} = 1 \ \forall s \leq t$, where $\tau_{ij}(t)$ is the travel time on arc (i, j) at time t.

[2]The modified label-correcting algorithm for the classic shortest path problem is defined in AHUJA, MAGNANTI, and ORLIN (1993) as a general label correcting algorithm with a scan-eligible list of nodes.

costs and lower bounds on the expected costs of these paths, respectively. These modifications are required because the least expected cost cannot be determined by simply replacing the time-varying travel time random variables by attributes others than travel time. This point was noted by ORDA and ROM (1991) and ZILIASKOPOULOS and MAHMASSANI (1992) in the context of least cost paths in deterministic, time-varying networks.

The primary contributions of this paper include derivation of the rationale and design of the specific computational steps to find the a priori least expected time paths with the associated expected times, lower bounds on the least expected times, the least expected time hyperpaths for the time-adaptive route choice problem, and the modifications to these procedures to determine a priori least expected cost paths and hyperpaths (with associated expected costs) and lower bounds on the expected costs of these paths from all origins to a single destination for each departure time in a given period in STV networks. The efficient procedure for determining lower bounds on the a priori least expected times (costs) provides valuable insight into the trade-offs between solution content (i.e., what one is seeking to determine) and the corresponding efficiency of the solution procedure. Furthermore, extensive numerical experiments are conducted to assess the computational performance of both procedures, and to compare the quality of the resulting lower bound in the second procedure.

The next section presents the EV algorithm, followed, in Section 2, by the ELB algorithm. Both algorithms and the manner in which they interrelate are discussed in Section 3. The numerical experiments are discussed in Section 4, followed by concluding comments in Section 5.

1. LEAST EXPECTED TIME PATHS

IN THIS SECTION, the problem definition and mathematical formulation of the problem of determining LET paths in STV networks are given, followed by the rationale and algorithmic steps of the EV algorithm.

1.1 Problem Definition and Mathematical Formulation

Let $G = (\mathcal{V}, \mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{P})$ be a directed graph where $\mathcal{V}$ is the set of nodes, $|\mathcal{V}| = v$, and $\mathcal{A}$ is the set of arcs, $|\mathcal{A}| = m$. Travel times along the arcs are represented by discrete random variables with distribution functions that are time varying over the period of interest, $t_0 \leq t \leq t_0 + (I)\delta$, referred to as the "peak period," and are stationary any time thereafter, $t > t_0 + (I)\delta$; the network is considered at a set $\mathcal{S}$ of discrete times $\{t_0 + n\delta\}$, where n is an integer, $n = 0, 1, 2, \ldots, I$, and δ is the smallest increment of time over which a perceptible change in the travel time distributions will occur for $t \in \mathcal{S}$.[3] This formulation can be generalized to travel times with continuous distributions.

For each departure time $t \in \mathcal{S}$ and each arc $(i, j) \in \mathcal{A}$, $\mathcal{T}_{i,j}(t)$ is the set of non-negative real valued possible travel times $\tau^k_{i,j}(t)$ for traversing the arc at the time t, $k = 1, \ldots, K_{i,j}(t)$, where $K_{i,j}(t)$ is the number of possible travel time values on arc (i, j) at time t. Travel time $\tau^k_{i,j}(t)$ occurs with probability $\rho^k_{i,j}(t)$ and $\Sigma_{k=1}^{K_{i,j}(t)} \rho^k_{i,j}(t) = 1$, $\forall t \in \mathcal{S}$. It is assumed that $\tau^k_{i,j}(t) = \tau^k_{i,j}(t_0 + I\delta)$ and $\rho^k_{i,j}(t) = \rho^k_{i,j}(t_0 + I\delta)$ $\forall k = 1, \ldots, K_{i,j}(t)$ and $(i, j) \in \mathcal{A}$, for all t occurring after the peak period, i.e., $\forall t > t_0 + I\delta$. Both the arc travel times and associated probability of occurrence are assumed to be given $\forall t \in \mathcal{S}$ and each arc $(i, j) \in \mathcal{A}$.

Let $\theta^c_i(t)$ be the travel time random variable for the cth path from node i to the destination node N at departure time interval t. For each $t \in \mathcal{S}$, the path with the minimum expected time, $E[\theta^c_i(t)]$, is sought. In other words, the problem is to determine the LET path from each node $i \in \mathcal{V}\backslash N$ to the given destination N, for each departure time $t \in \mathcal{S}$, in a network where the arc travel times are given by time-dependent, random variables with probability distribution functions that are known a priori. Furthermore, it is assumed that the arc travel time random variables are independent across arcs and over time, and no waiting is permitted at any intermediate node. That is, one must leave an intermediate node immediately upon arrival.

The EV algorithm is a specialized modified label-correcting algorithm for generating the a priori LET paths. In the generic label-correcting algorithm for time-invariant, deterministic shortest path problems, a single label associated with each node maintains the current shortest distance from the node to the destination. The labels are updated iteratively until the optimality conditions, based on Bellman's principle of optimality (which states that a path between any pair of nodes on a shortest path in a given network must itself be a shortest path, [BELLMAN, 1958]), are satisfied. Bellman's principle of optimality also holds on the space–time representation of networks with deterministic, time-dependent arc weights (ZILIASKOPOULOS and MAHMASSANI,

[3]The size of the time interval δ must be smaller than any travel time on an arc; otherwise, pathological inconsistencies may occur because it would be possible to arrive at the next node en route at the same time as the departure time from the origin.

1993). However, it does not apply directly to networks where the arc weights are time-dependent random variables. An extension of this principle, given in Proposition 1, is required for this case because the expected path times are not simply the sum of the expected arc times. In deterministic networks, a path is shorter than (dominates) another if its label value is lower than the other's label value. However, in STV networks, conditions for dominance over a time period, hereafter referred to as EV-dominance, must be established, as follows. A path c is nondominated iff $\exists$ no path d such that

$$\mathrm{E}[\theta_i^d(t)] \leqslant \mathrm{E}[\theta_i^c(t)] \quad \forall t \in \mathcal{S}$$

$$\text{and} \quad \exists t \in \mathcal{S} | \mathrm{E}[\theta_i^d(t)] < \mathrm{E}[\theta_i^c(t)];$$

otherwise, the path is dominated. Given these EV-dominance conditions, we extend Bellman's principle of optimality as follows.

PROPOSITION 1. *All subpaths of a nondominated path with the same destination node as this path must themselves be nondominated.* (*This can be generalized to all subpaths of a nondominated path in FIFO networks.*)

A formal proof of this proposition can be constructed along the lines of HANSEN (1980) and is given in MILLER-HOOKS (1997).

1.2 Solution Approach

For each node $i \in \mathcal{V}$ and each potentially optimal path c to the destination node N, a vector label $[\lambda_i^c(t)]_{t \in \mathcal{S}}$ is maintained, denoted Λ_i^c, where $\lambda_i^c(t)$ is the expected travel time along path c from node i to the destination, leaving node i at time t; i.e., $\lambda_i^c(t) = \mathrm{E}[\theta_i^c(t)]$. These labels are called p-optimal because each is potentially optimal for one or more time intervals. Until termination of the algorithm, more than one label vector is maintained at each node unless a single label is best for all time intervals. Let $\chi(i)$ be the set of p-optimal labels at node i. A scan eligible (SE) list of labels Λ_j^d, identified by the node–label pair $(j\text{–}d)$ (which uniquely identifies a distinct path), is maintained. At each iteration of the algorithm, a label Λ_j^d is selected from the SE list.[4] A temporary label vector is constructed, $\kappa_i = [\kappa_i(t)]_{t \in \mathcal{S}}$, from each predecessor node i of node j $(i \in \Gamma^{-1}(j))$. To determine if it is p-optimal, it is compared with the p-optimal labels at node i, Λ_i^c (according to the optimality conditions given in step 2 of the algorithm below) and if it is determined to be dominated, the temporary path is discarded.[5]

When the algorithm terminates, with the exception of ties (broken arbitrarily), a single best path will be associated with each time interval for each node $i \in \mathcal{V}$. Because the same path may be optimal for more than one time interval, at most I (the number of time intervals in the peak period) optimal paths can be in the final solution set for each node.

Two pointers are required for each label c at each node i to store the p-optimal paths efficiently: a pointer, π_i^c, from the cth label at node i to the next node on the path and a pointer, L_i^c, to indicate the appropriate label at the next node. Note that π_i^{temp} and L_i^{temp} hold the path information of a temporary label until that label is determined to be p-optimal or is discarded.

1.3 The EV Algorithm

The steps of the algorithm are described hereafter.

Step 0: Initialization and Creation of the Scan Eligible List

Initialize the node labels:

Initialize labels and path pointers

$$\lambda_i^c(t) = \infty$$

$$\forall i \in \mathcal{V},\ c \in \{1, 2, \ldots, M\}, \quad t \in \mathcal{S},$$

where M is a large enough number to permit as many p-optimal paths at any node as might be required.

$$\lambda_N^1(t) = 0 \quad \forall\, t \in \mathcal{S}.$$

$$\pi_i^c = \infty \quad \text{and} \quad L_i^c = \infty$$

$$\forall c \in \{1, 2, \ldots, M\} \quad \text{and} \quad i \in \mathcal{V}.$$

$\chi(N) = \{1\}$ (put the first label at node N in the set of p-optimal labels from N).

Initialize the scan eligible list:

Insert node–label pair N-1 in the SE list.

Step 1: Check List and Scan Node

If the SE list is not empty, select the first node–label pair $(j\text{-}\mu)$ from the front of the list. Call the associated node j the current node. If the list is empty, go to Step 3.

[4]It is assumed that $\forall i \in \mathcal{V}$; $c \in \chi(i)$, $t \in \mathcal{S}$, and $0 < \varepsilon < \delta$, $\lambda_i^c(t + \varepsilon) = \lambda_i^c(t)$ and that for all t occurring outside the peak period, $t > t_0 + I\delta$, $\lambda_i^c(t) = \lambda_i^c(t_0 + I\delta)$.

[5]If a temporary label is dominated by any currently p-optimal path, it cannot dominate any remaining p-optimal path. Similarly, if the temporary label dominates any p-optimal path, then it cannot be dominated by any other currently p-optimal path. These facts can be used to reduce the number of comparisons.

Step 2: Update Node Labels

For each $i \in \Gamma^{-1}(j)$ (i.e., $\forall i|(i, j) \in \mathcal{A}$):

Temporary Label Creation

Determine the expected time $\kappa_i(t)\ \forall t \in \mathcal{S}$ for the newly constructed path from node i through Λ_j^μ by the following equation:

$$\kappa_i(t) = \sum_k ([\tau_{i,j}^k(t) + \lambda_j^\mu(t + \tau_{i,j}^k(t))] \cdot \rho_{i,j}^k(t)), \quad (1)$$

$$\text{where } k = 1, 2, \ldots, K_{i,j}(t).$$

(More detail on how to construct the temporary label is given in Appendix A).

Set the path pointers: $\pi_i^{\text{temp}} = j$ and $L_i^{\text{temp}} = \mu$.

Label Comparisons

Compare κ_i with each Λ_i^c, $c \in \chi(i)$:

$[\kappa_i(t)]_{t\in\mathcal{S}}$ is p-optimal iff $\exists$ no path $c \in \chi(i)$ such that $\lambda_i^c(t) \leqslant \kappa_i(t)\ \forall t \in \mathcal{S}$ and $\exists t \in \mathcal{S}|\lambda_i^c(t) < \kappa_i(t)$; otherwise, $[\kappa_i(t)]_{t\in\mathcal{S}}$ is dominated—discard $[\kappa_i(t)]_{t\in\mathcal{S}}$.

If a path has the least expected time of all p-optimal paths for a given departure time, that path can be marked as the LET path for the associated time interval in this step.

If $[\kappa_i(t)]_{t\in\mathcal{S}}$ is p-optimal, add to $\chi(i)$ and put this node-label pair in the SE list.

Check if all $c \in \chi(i)$ are still p-optimal and remove the non-p-optimal labels from $\chi(i)$.[6]

If all $i \in \Gamma^{-1}(j)$ have been scanned, go to Step 1.

Step 3: Stop.

The least expected time paths with their expected times are given, for each departure time interval, by each label $c \in \chi(i)$ that is marked as having the least expected time, and its associated path pointers. Only one path in $\chi(i)\ \forall i \in \mathcal{V}$, for each $t \in \mathcal{S}$, will be marked (ties broken arbitrarily). Thus, the algorithm terminates with the set of least expected time paths and the associated expected times $\forall t \in \mathcal{S}$ from every node $i \in \mathcal{V}$ to the destination node N.

The SE list can be implemented in several ways. The description of the algorithm suggests that a FIFO SE list is used; however, a deque structure is equally viable (AHUJA, 1989; GLOVER et al., 1985). In the above description, node–label pairs are inserted in the list. Because it is possible that the algorithm's performance could improve if the labels from the same node are consecutively scanned, one may consider maintaining the SE list as a list of nodes, where each node can enter at most one time with a separate list of labels for each node. When a node is selected from the SE list, each label in this list is scanned before the next node is selected. A similar implementation was tested by ZILIASKOPOULOS (1992) for the K-shortest path problem (with deterministic, time-invariant travel times) and by Miller-Hooks and Mahmassani (1998) for determining least possible time paths in stochastic, time-dependent networks. For more information on structuring the scan eligible list, see PAPE (1974), PALLOTTINO (1984) and GALLO and PALLOTTINO (1986).

This algorithm is easily extended for determining least expected cost paths in networks where both the arc travel times and arc costs are random variables with time-varying probability distribution functions. Thus, the network G is expanded to $G = (\mathcal{V}, \mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{P}_\mathcal{T}, \mathcal{C}, \mathcal{P}_\mathcal{C})$; $(\mathcal{C}, \mathcal{P}_\mathcal{C})$ is the set of non-negative real valued possible arc costs $\forall (i, j) \in \mathcal{A}$: $C_{i,j}^x(t)$, $x = 1, \ldots, X_{i,j}(t)$, where $X_{i,j}(t)$ is the number of possible cost values of travel times on arc (i, j) at time t. Cost $C_{i,j}^x(t)$ occurs with probability $q_{i,j}^x(t)$ and $\Sigma_{x=1}^{X_{i,j}(t)} q_{i,j}^x(t) = 1\ \forall t \in \mathcal{S}$. Similar to arc travel times, it is assumed that $C_{i,j}^x(t) = C_{i,j}^x(t_0 + I\delta)$ and $q_{i,j}^x(t) = q_{i,j}^x(t_0 + I\delta)\ \forall t > t_0 + I\delta$. The cost and travel time are assumed to be independent of one another. The labels, Λ_i^c, $\forall i \in \mathcal{V}$, $c = 1, \ldots, M$, now maintain the expected cost of the paths with the least expected cost for at least one time interval. Equation 1 of Step 2 for computing the expected cost of a newly constructed path from node i at departure time t can be replaced by equation (1′),

$$\kappa_i(t) = \sum_k \sum_x ([c_{i,j}^x(t) + \lambda_j^\mu(t + \tau_{i,j}^k(t))] \cdot q_{i,j}^x(t) \cdot \rho_{i,j}^k(t)), \quad (1)$$

where $k = 1, 2, \ldots, K_{i,j}(t)$, $x = 1, 2, \ldots, X_{i,j}(t)$.

PROPOSITION 2. *The EV algorithm terminates with the set of nondominated solutions. No path can exist that dominates any of these paths.*

Proof. Upon termination, no label Λ_i^c corresponding to a nondominated path exists for which the following relation holds (for some label Λ_i^d):

$$\lambda_i^d(t) \leqslant \lambda_i^c(t) \quad \forall t \in \mathcal{S} \quad \text{and} \quad \exists t \in \mathcal{S}|\lambda_i^d(t) < \lambda_i^c(t) \quad (2)$$

otherwise, label Λ_i^c would be dominated.

[6]The fact that a nondominated path cannot contain a dominated subpath to the destination node (by Proposition 1) can be used in the implementation of the algorithm to aid in decreasing the run times. If a path is known to be dominated, its node–label pair can be placed in the SE list and marked as dominated. When this node–label pair is chosen from the SE list, all existing paths from predecessor nodes of the origin of this dominated path can also be marked as dominated and their node–label pairs can be added to the SE list. The details of this implementation are given in Miller-Hooks (1997).

Suppose there exists a label Λ_i^d such that Eq. 2 holds. Then, $\exists j \in \Gamma^{+1}(i)$ and Λ_j^e, a subpath of Λ_i^d, such that

$$\lambda_i^c(t) \geqslant \sum_{k=1}^{K_{i,j}(t)} \{[\tau_{i,j}^k(t) + \lambda_j^e(t + \tau_{i,j}^k(t))] \cdot \rho_{i,j}^k(t)\} \quad \forall t \in \mathcal{S}$$

and

$$\exists t \in \mathcal{S} | \lambda_i^c(t) > \sum_{k=1}^{K_{i,j}(t)} \{[\tau_{i,j}^k(t) + \lambda_j^e(t + \tau_{i,j}^k(t))] \cdot \rho_{i,j}^k(t)\}.$$

Then either Λ_i^d is dominated by another path or node–label pair j–e has not been scanned. If Λ_i^d is dominated by another path, this other path would also dominate Λ_i^c. Thus, node–label pair j–e must be in the SE list. This contradicts the statement that the algorithm has terminated. No path can exist that dominates one of the final nondominated solutions determined by the EV algorithm. □

PROPOSITION 3. *The EV algorithm terminates after a finite number of steps.*

Proof. To prove that the EV algorithm terminates after a finite number of steps is equivalent to proving that the SE list is empty after a finite number of steps. Suppose the SE list is not empty in a finite number of steps. Then, either (1) at least one node–label pair (representing a distinct path) must be inserted in the SE list an infinite number of times, or (2) an infinite number of node–label pairs must enter the SE list. Case (1) is not possible because each node–label pair (indicating a particular path), can only enter the SE list at most twice (once when it first becomes p-optimal and once later when (if) it is determined to be dominated, if this option is implemented). Thus, a node–label pair cannot enter the SE list an infinite number of times. Case (2) is not possible because the graph is assumed to be finite (contains v nodes and m arcs). Thus, a finite number of simple (acyclic) paths exist. Furthermore, an infinite number of paths with cycles cannot enter the SE list because the arcs have positive, real-valued weights. If a path indicated by a node–label pair enters the SE list (and was not entered as a result of becoming dominated), its expected travel time must be lower than that of another p-optimal path for at least one departure time interval and must not be dominated by any other p-optimal path. Given that there are a finite number of departure time intervals in $\mathcal{S}$ and that the arc travel times are positive real-values, this can occur only a finite number of times. This contradicts the assumption that the SE list is not empty in a finite number of steps. □

The actual number of paths that may have the least expected time (cost) for one or more departure time intervals must be no greater than I, where I is the number of time intervals in $\mathcal{S}$, because at most one path has the least expected time (cost) for each departure time (ties broken arbitrarily). However, an arbitrarily large (but finite) number of labels may need to be maintained at each node if $I > 1$ because the algorithm must determine all EV-nondominated paths to guarantee that it will generate all a priori LET paths (Miller-Hooks, 1997). In Proposition 4, it is shown that the number of such nondominated paths may grow exponentially with the network size; as such, the algorithm has similar worst-case performance to that of solution methodologies for bicriterion path problems. A proof is given by Hansen (1980) of the intractability of labeling algorithms for determining all nondominated paths for the bicriterion shortest path problem. This is further discussed in BRUMBAUGH-SMITH and SHIER (1989).

PROPOSITION 4. *The EV algorithm may result in an exponentially growing number of nondominated solutions with the network size if* $I > 1$.

Proof. Assume $I = 2$. A label for every possible path from a node to the destination node may need to be maintained because it is possible that no path has a lower expected time than another path for every time interval $t \in \mathcal{S}$. Assume $e > 0$ and $x > y$, then the following may occur:

	Path 1	Path 2	Path 3	Path 4	⋯
Time = 1	x	$x + e$	$x + 2e$	$x + 3e$	⋯
Time = 2	y	$y - e$	$y - 2e$	$y - 3e$	⋯

No path listed above is better than any other for both time intervals. Thus, all paths must be maintained. This applies to $I > 2$. □

Because an arbitrarily large number of paths may be maintained at each node, although finite, this algorithm can perform, in the worst-case, very poorly—nonpolynomially. However, as is shown in numerical experiments in Section 4, the average performance of this algorithm for networks with an average in- and out-degree of 4 from each node is much better than predicted by the worst-case computational complexity analysis. Similar results were shown by Brumbaugh-Smith and Shier (1989) with respect to the performance of label-correcting procedures for finding all nondominated solutions to the bicriterion shortest path problem in deterministic networks.

2. LOWER BOUND ON LEAST EXPECTED TIME PATH

IN THIS SECTION, an efficient algorithm, referred to as the ELB algorithm, is presented for determining lower bounds on the expected times of the least expected time paths in $G = (\mathcal{V}, \mathcal{A}, \mathcal{S}, \mathcal{T}, \mathcal{P})$ defined in Section 1.1, $\forall i \in \mathcal{V}$ to a single destination, $\forall t \in \mathcal{S}$. This lower bound can provide a benchmark against which to evaluate one or more paths that may have been generated using a heuristic (but efficient) path search procedure. With minor modification, the ELB algorithm also provides an exact solution in the form of hyperpaths to the problem of determining the least expected time paths from all origins to a given destination for each departure time $t \in \mathcal{S}$, when the driver can select the next arc upon arrival at a node, i.e., after experiencing the revealed (actual) travel times on traveled arcs while en route (i.e., in a time-adaptive route choice framework). The modifications are given parenthetically in the algorithm description.

The ELB algorithm, like the EV algorithm, is a specialized modified label-correcting algorithm. Here, a single label vector, $[\lambda_i(t)]_{t\in\mathcal{S}}$, is associated with every node. At termination of the algorithm, each $\lambda_i(t)$ is a lower bound on the a priori least expected time for any path to the destination node for time t. Before termination of the algorithm, $\lambda_i(t)$ remains an upper bound to the desired value. However, these labels do not necessarily correspond to individual paths and paths cannot be reconstructed upon termination. To track the solution hyperpaths for the time-adaptive route choice problem, a second label vector can be introduced, $[\pi_i(t)]_{t\in\mathcal{S}}$, where $\pi_i(t)$ indicates the arc to be followed from node i and time t. The SE list used in this algorithm consists of a list of nodes.

Let the vector $[\eta_i(t)]_{t\in\mathcal{S}}$ be the temporary label from node i. Denote by $\lambda_i^k(t)$ the current label component, associated with departure time t, at the end of the kth iteration. At the $k+1$th iteration, the optimality conditions can be stated as follows: $\lambda_i^{k+1}(t) = \min\{\lambda_i^k(t), \eta_i(t)\}$ for each $t \in \mathcal{S}$. The steps of the ELB algorithm are presented next.

Step 0. Initialization

Initialize the node labels:

$\lambda_i(t) = \infty \quad \forall i \in \mathcal{V}\backslash N, \quad t \in \mathcal{S}.$

$(\pi_i(t) = \infty \quad \forall i \in \mathcal{V}\backslash N, t \in \mathcal{S}).$[7]

$\lambda_N(t) = 0 \quad \forall t \in \mathcal{S}.$

Initialize the scan-eligible list:

Create the scan-eligible list, SE, and insert N.

Step 1. Choose Current Node

If the SE list is empty, go to step 3.

Otherwise, select the first node from the SE list.

Call this node the current node, j.

Step 2. Update the Node Labels

For each $i \in \Gamma^{-1}(j)$, (i.e., $\forall i|(i,j) \in \mathcal{A}$), update the vector $[\lambda_i(t)]_{t\in\mathcal{S}}$ as follows:

For each $t \in \mathcal{S}$:

Calculate

$$\eta_i(t) = \sum_{p=1}^{K_{i,j}(t)} [(\tau_{ij}^p(t) + (\lambda_j(t + \tau_{ij}^p(t))) \cdot \rho_{ij}^p(t)], \tag{3}$$

where p is the set of indices of possible travel times on arc (i,j) at time t.

If $\eta_i(t) < \lambda_i(t)$, then $\lambda_i(t) = \eta_i(t)$, $(\pi_i(t) = (i,j))$[8] and if $i \notin$ SE, put i in SE list.

If all $i \in \Gamma^{-1}(j)$ have been considered, go to step 1.

Step 3. Stop

The algorithm terminates with a lower bound on the expected time of the a priori LET paths, $\forall t \in \mathcal{S}$, from every node $i \in \mathcal{V}$ to the destination node N.

PROPOSITION 5. *The final solution provides a lower bound on the a priori LET for any path from each node in $\mathcal{V}$ to destination node N, $\forall t \in \mathcal{S}$.*

Proof. Let $\{E[\theta_j^c(t)]\}_{t\in\mathcal{S}}$ be the vector of expected times associated with the cth path from node j to N and $[\lambda_j(t)]_{t\in\mathcal{S}}$ defined previously. Trivially, $[\lambda_N(t)]_{t\in\mathcal{S}} = \{E[\theta_N^c(t)]\}_{t\in\mathcal{S}}$ because it is assumed that there is only one path from the destination node to itself (with zero travel time for every departure time).

Consider the construction of a label from an origin node j, where there are several possible paths to the destination node. Suppose path c has the lowest expected time of all paths at time t_1, and path d has the lowest expected time of all paths at time t_2, both paths from node j, $\lambda_j(t_1) = E[\theta_j^c(t_1)]$ and $\lambda_j(t_2) = E[\theta_j^d(t_2)]$. Now, suppose a label is constructed from node i through node j. Suppose also that, for time t_0, there are two possible travel times, $\tau_1 = \tau_{i,j}^1(t_0)$ and $\tau_2 = \tau_{i,j}^2(t_0)$ with associated probabilities of occurring of $\Pr\{\tau_1\} = \rho_{i,j}^1(t_0)$ and $\Pr\{\tau_2\} = \rho_{i,j}^2(t_0)$. The possible arrival times at node j are t_1 and t_2, respectively. Let path e be a path from node i to destination node N, constructed from arc (i,j) and subpath c from node j to N. Similarly, let path f be a path from i to N, constructed from arc (i,j) and subpath

[7] Additional label vector of hyperpath pointers required for the time-adaptive route choice problem.

[8] Modification required to address the time-adaptive route choice problem.

d from j to N. Then

$$\lambda_i(t_0) = (\tau_1 + \lambda_j(t_1))(\Pr\{\tau_1\}) + (\tau_2 + \lambda_j(t_2))(\Pr\{\tau_2\})$$

$$=(\tau_1 + \mathrm{E}[\theta_j^c(t_1)])(\Pr\{\tau_1\}) + (\tau_2 + \mathrm{E}[\theta_j^d(t_2)])(\Pr\{\tau_2\})$$

$$\leqslant(\tau_1 + \mathrm{E}[\theta_j^c(t_1)])(\Pr\{\tau_1\}) + (\tau_2 + \mathrm{E}[\theta_j^c(t_2)])(\Pr\{\tau_2\})$$

$$= \mathrm{E}[\theta_i^e(t_0)]$$

and is also

$$\leqslant(\tau_1 + \mathrm{E}[\theta_j^d(t_1)])(\Pr\{\tau_1\}) + (\tau_2 + \mathrm{E}[\theta_j^d(t_2)])(\Pr\{\tau_2\})$$

$$= \mathrm{E}[\theta_i^f(t_0)].$$

That is, $\lambda_i(t_0) \leqslant \mathrm{E}[\theta_i^e(t_0)]$ and $\lambda_i(t_0) \leqslant \mathrm{E}[\theta_i^f(t_0)]$. This can be extended $\forall t \in \mathcal{S}$. Note that, if for some subpath m, $\lambda_j(t_1) = \mathrm{E}[\theta_j^m(t_1)]$ and $\lambda_j(t_2) = \mathrm{E}[\theta_j^m(t_2)]$, then $\lambda_i(t_0)$ is constructed through only one path, m, and $\lambda_i(t_0) = \mathrm{E}[\theta_i^n(t_0)]$, where path n is constructed from arc (i, j) and subpath m from j to N. Any $\lambda_i(t)$ may be constructed from either a single path or from the best values of several paths; and therefore, $\lambda_i(t) \leqslant \mathrm{E}[\theta_i^g(t)]$ (for any path g from i to N). This can be extended to all possible origins in the network. Thus, for each $t \in \mathcal{S}$, the label vector $[\lambda_i(t)]_{t\in\mathcal{S}}$ gives a lower bound on the expected time of the a priori LET path. □

Note that the final label for any node is exactly the least expected time for a path to the destination node if only one path from that node contributes to its label. Extensive computational experiments, described in Section 4, suggest that a high percentage of solutions are composed of travel times from only a single path. For such solutions, actual paths can be identified. An example is shown in Appendix B to illustrate the ELB algorithm and to clarify Proposition 5.

Similar to the adaptation of the EV algorithm presented in Section 1.3, the ELB algorithm can be modified to determine lower bounds on a priori least expected path costs. Let the label vector $[\lambda_i(t)]_{t\in\mathcal{S}}$, $\forall i \in \mathcal{V}$, now denote the lower bounds on the cost of least expected cost paths from node i to the destination at time t. Equation 3 of Step 2 can be replaced by

$$\eta_i(t) = \sum_k \sum_x [(c_{ij}^x(t) + (\lambda_j(t + \tau_{ij}^k(t))) \cdot q_{ij}^x(t) \cdot \rho_{ij}^k(t)] \tag{3'}$$

where $k = 1, 2, \ldots, K_{i,j}(t)$, $x = 1, 2, \ldots, X_{i,j}(t)$.

Note that the ELB algorithm provides an upper bound on the expected least time, $\mathrm{E}[\min_c \theta_i^c(t)]$, through the network, i.e., the wait-and-see bound of MADANSKY (1960). This measure would be obtained by taking the expectation, over many realizations, of the minimum travel time path through the network for a given joint realization of all arc weights.

PROPOSITION 6. *The ELB algorithm terminates in a finite number of steps.*

Proof. The algorithm terminates in a finite number of steps if the SE list is empty in a finite number of steps. Suppose the SE list is not empty in a finite number of steps, then at least one node must be inserted in the SE list an infinite number of times. This implies that the label at the node has improved by at least a positive real-value of travel time. If the improvement at the node continues an infinite number of times, then the travel time on the path would eventually become negative, which contradicts the assumption of positive travel times. This contradicts the supposition that the SE list is not empty in a finite number of steps and hence shows that the ELB algorithm terminates in a finite number of steps. □

The next proposition establishes that the ELB algorithm determines the exact least expected time (cost) from every origin to N for each departure time interval $t \in \mathcal{S}$ for the time-adaptive route choice problem. To obtain the associated hyperpaths (optimal strategies), an additional label vector must be updated, as noted in the algorithm description, (see Appendix B for an example problem).

PROPOSITION 7. *The ELB algorithm terminates with the set of least expected time (cost) paths (i.e., hyperpaths) for the time-adaptive route choice problem.*

Proof. Upon termination, the following relation holds for every label at every $t \in \mathcal{S}$:

$$\lambda_i(t) \leqslant \sum_{k=1}^{K_{i,j}(t)} \{[\tau_{i,j}^k(t) + \lambda_j(t + \tau_{i,j}^k(t))] \cdot \rho_{i,j}^k(t)\}$$

$$\forall j \in \Gamma^{+1}(i),$$

where $\Gamma^{+1}(i)$ represents the set of successor nodes of node i (i.e., $j|(i, j) \in \mathcal{A}$). Suppose, at termination, a component of a label vector, $\lambda_i(t)$, exists such that

$$\lambda_i(t) > \sum_{k=1}^{K_{i,j}(t)} \{[\tau_{i,j}^k(t) + \lambda_j(t + \tau_{i,j}^k(t))] \cdot \rho_{i,j}^k(t)\}$$

for some $j \in \Gamma^{+1}(i)$, then j was not scanned and hence must be in the SE list. This contradicts the assumption of termination, and, hence, the assumption that such a label exists. Therefore, the algorithm terminates with the minimum label for each node at each $t \in \mathcal{S}$.

To show that this label corresponds to the LET hyperpath for the time-adaptive route choice problem from the associated node to the destination for the given departure time, we proceed by induction.

Let k be the number of arcs on a path to the destination node. If k is zero, then the path to the destination is trivial and obviously leads to the LET path for every departure time. For arbitrary k, we assume that the label from a given node j for departure time t gives the LET path for this problem by providing the first arc (j, h) on the path to N. For $k + 1$, the label for a given node $i \in \Gamma^{-1}(j)$, for departure time t, provides the arc (i, j) such that j is k arcs from n (given i is $k + 1$ arcs from n). Suppose there are $K_{i,j}(t)$ possible arc travel times for departure time t, $\tau_{i,j}^1(t)$, $\tau_{i,j}^2(t)$, . . . , $\tau_{i,j}^{K_{i,j}(t)}(t)$, then there are $K_{i,j}(t)$ possible arrival, and, hence, departure times from node j: $[t + \tau_{i,j}^k(t)]_{k=1,\ldots,K_{i,j}(t)}$. After the driver traverses (i, j), the travel time along the arc is no longer uncertain and the exact departure time from node j is known. For the known departure time, the corresponding arc on which to proceed (to follow the least expected time path in this framework) is given from node j, which is k arcs from N. Therefore, $k \rightarrow k + 1$ for arbitrary k and, hence, it follows by mathematical induction that the label and associated arc provides the least expected time and associated hyperpath for the time-adaptive route choice problem. □

Proposition 8 establishes that the ELB algorithm has similar worst-case computational complexity to that of the time-dependent least time algorithm presented by Ziliaskopoulos and Mahmassani (1993) ($\sim$O(I^2v^3)) for deterministic, time-dependent networks.

PROPOSITION 8. *The ELB algorithm with a basic FIFO SE list structure has worst-case computational complexity* $\sim$O(I^2v^3P), *where I is the number of time intervals into which the peak period is discretized*, *v is the number of nodes in the network*, *and P is the maximum number of possible values of the discrete arc travel time random variable for a time interval.*

Proof. After the destination node is removed from the SE list, it will never again be updated; i.e., it is permanently set. All of its predecessor nodes are added to the SE list for updating. Thus, the SE list contains at most $v - 1$ nodes. From all nodes initially inserted in the SE list (i.e., predecessors of the destination node), the one with the least label for a given departure time will be updated permanently. By scanning all nodes (at most $v - 1$) of the SE list in as many as $v - 1$ repetitions of Step 2, at least one label will be permanently set. This procedure may be repeated at most $(I)(v - 1)$ times because there are $(I)(v - 1)$ label components in total that can be improved. Thus, there are at most $(I)(v - 1)^2$ repetitions of Step 2. Step 2 requires a maximum of $P(I)(v - 1)$ $\sim$O(1) computations, because, in the worst-case, each node can be reached by $v - 1$ nodes and each node has I labels, requiring P computations. This results in worst-case computational complexity of $\sim$O($(I)(v - 1)^2 \cdot (P)(I)(v - 1)$), or $\sim$O(I^2v^3P). □

3. DISCUSSION OF RATIONALE FOR THE EV AND ELB ALGORITHMS

THIS SECTION EXPLAINS why is it more difficult to determine the LET path in STV networks than it is to find the least time paths in deterministic, time-dependent networks, and clarifies some of the differences between the EV and ELB algorithms. The intent is to provide the reader with deeper insight into the complexity of the problem, why and how the algorithms presented work, as well as the trade-off between computational burden and desired solution.

In deterministic, time-varying networks a single scalar label is associated with each departure time interval at a node, representing the travel time to the destination. Say a path is constructed from node i through node j, via arc (i, j). Given the labels at node j for all time intervals, a label at node i for departure time t can be constructed by adding the travel time of arc (i, j) for the respective departure time to the label at node j corresponding to the appropriate (single) arrival time. The path tree can be maintained through two pointers from each node at each time interval: a pointer to node j and another to the arrival time interval. See Ziliaskopoulos and Mahmassani (1993) for more information on solving the deterministic, time-dependent least time path problem.

Assume now that the travel time on arc (i, j) is a random variable with given probability density or mass function. In such a network, arrival time at node j from i depends on the actual travel time on arc (i, j). Assume there are two possible arrival times, s and q. If the LET labels at node j at times s and q correspond to different paths, then it will no longer be possible to maintain a single path label for node i at time t (hence the hyperpath structure of the solution for the time-adaptive route choice problem). If only a single label is maintained, it can give a lower bound on the least expected time that may be obtained via either subpath from node j at time intervals s and q. One can no longer maintain the paths in a shortest path tree structure. To continue to maintain the least expected times and their associated paths, it will be necessary to maintain more than one label at node i for time interval t. This is the essential concept underlying the EV algorithm. Unfortunately, it is, in part, for this same reason that the EV algorithm is theoretically inefficient. In

the next section, results are given from a series of experiments conducted on randomly generated networks to examine the actual average performance of the algorithms.

4. NUMERICAL EXPERIMENTS

THE PERFORMANCE OF both the EV and ELB algorithm is examined through numerical experiments on randomly generated networks with average in- and out-degree each of four; additional tests are also conducted on networks with higher average degree. The methodology for generating the networks with their STV arc weights is described and the experimental design is given. The results of the tests are presented and analyzed.

The specific hypotheses investigated through these experiments are as follows.

1. The increase in actual (average-case) run time, as the network size increases is better than predicted through worst-case computational complexity for both the EV and ELB procedures.
2. The number of EV-nondominated paths from any node is small.
3. The number of labels required to determine these paths does not grow exponentially with the size of the network.
4. The lower bounds on the least expected times determined by the ELB procedure are close to the expected times of the true least expected time paths.
5. The EV-nondominated paths determined by the EV algorithm are robust in the sense that they have the least expected time for more than one departure time interval.

To test these hypotheses, the EV and ELB procedures are evaluated in terms of the following applicable performance characteristics: actual run times; maximum and average numbers of EV-nondominated paths from any origin; number of pairwise comparisons; percent relative difference; maximum and average percent relative difference of the ELB algorithm lower bounds from the true least expected times; and number of time intervals an EV-nondominated path has the least expected time.

4.1 Experimental Design

The experiments described in this section are conducted on twelve randomly generated networks with randomly generated time-varying probability mass functions (PMFs) of the arc travel time random variables, as described hereafter.

4.1.1 Generating the Networks and Arc Travel Time Random Variables

A random network generator (RNG) adapted from ZILIASKOPOULOS (1994) is used for these experiments. The number of nodes is prespecified, the arcs are directed and are uniformly randomly generated. The procedure used to ensure connectivity is detailed in Miller-Hooks (1997). The networks were generated such that their average in-degree and out-degree at a node are each approximately 4 (between 3.6 and 4.8), consistent with the application to transportation systems. The in- or out-degree of each node in these networks can vary between 2 and 9. The results of another set of experiments on networks with higher average degrees are discussed later in this section. The relatively constant average degree ensures that networks with the same number of nodes will have nearly the same number of arcs.

Given a specified network topology, the PMFs of the independent arc weight random variables, for each $t \in \mathcal{S}$, were randomly generated. The PMFs correspond to either discrete random variables or approximations of continuously distributed random variables. The number of elements in the PMFs, P, is assumed constant across arcs and departure times. For each departure time interval, for each directed arc, P pairs of scaled uniform random variates are generated; the first corresponding to a possible travel time and the second to the probability of the occurrence of such a travel time. These are then normalized such that the sum of the second random variates is one. The values are sorted by value of the first random variate and the probabilities associated with identical travel times are added to produce the PMF for one departure time.

4.1.2 Design of the Experiments

Three factors must be specified to generate the network topology and the PMFs of the arc travel time random variables: the number of nodes, duration of the peak period (i.e., number of time intervals), and the number of elements in the PMFs. Four levels of the number of nodes are considered: 50, 100, 500, and 1000 nodes. Three topological networks of each level are generated, for a total of 12 networks. The time interval size is one time unit in duration and is constant over all the experiments. Four levels of the duration of the peak period are considered: 10, 30, 60, and 90 time intervals. Finally, three levels of the constant number of elements in the PMFs are considered: 5, 10, and 20. This results in 144 different combinations, because every combination of the number of time intervals

and number of elements in the PMFs are considered for each of the 12 networks. The three networks with the same number of nodes are described by the (n, t, p)-triple corresponding to n nodes, t time intervals, and p elements in the arc weight PMFs.

The EV and ELB algorithms are implemented in FORTRAN and run on a DEC 600/5/266 AlphaStation with 256 megabytes ram, running under DEC UNIX 3.2C. For each run, a destination node is randomly selected. Thirty such destination nodes are chosen, resulting in 30 runs for each of the 144 combinations. A set of 30 runs for one of the 144 combinations is referred to as an experiment. A total of 144 experiments, consisting of 4320 runs, is conducted on both the EV and ELB procedures on the networks with average in- and out-degrees of four, resulting in 8640 runs. Additional runs are completed on two 500-node networks with average in- and out-degrees of 10, to estimate the effects of the average degree of the network on the closeness of the ELB to the true LET.

4.2 Principal Performance Measures Considered

The performance measures used to evaluate the EV and ELB procedures are described in this section. Each of these measures is used in evaluating the suggested hypotheses.

Run time in cpu seconds: The average user cpu[9] time required to run each procedure over the 30 destination nodes is measured for each (n, t, p)-triple for each of the three networks with n nodes. The run times do not include i/o time. All other steps of the procedures, including statements required to measure other performance characteristics, are included.

Average number of EV-nondominated paths: The average number of EV-nondominated paths over all origins and all 30 destinations is computed for each (n, t, p)-triple, for each of the three networks with n nodes.

Maximum number of EV-nondominated paths: For each (n, t, p)-triple, for each of the three networks with n nodes, the maximum number of EV-nondominated paths from any origin is determined. The average maximum value over the 30 destinations is computed.

Number of label comparisons: The number of pairwise label comparisons in Step 2 in the EV and ELB algorithms are measured. Each comparison consists of a set of comparisons over the time period $\mathcal{S}$. For each (n, t, p)-triple, for each of the three networks with n nodes, the average number of label comparisons over all origins and the 30 destinations is computed.

Percent relative difference: For each (n, t, p)-triple, for each of the three networks with n nodes, the percent relative difference is the difference between the true expected time of the LET path (obtained from running the EV algorithm) and the lower bound of the least expected time (obtained from running the ELB algorithm) for a given departure time and given origin divided by the lower bound obtained from the ELB procedure and multiplied by 100.

Maximum percent relative difference: For each (n, t, p)-triple, for each of the three networks with n nodes, the maximum percent relative difference from every node, for every departure time interval, and for each of the 30 destinations for the ELB lower bound is determined.

Average percent relative difference: For each (n, t, p)-triple, for each of the three networks with n nodes, the percent relative differences computed in Eq. 5 are averaged over all origins, all departure times, and the 30 destinations.

Number of time intervals for which the EV-nondominated paths have the least expected time: For each (n, t, p)-triple, for each of the three networks with n nodes, the average number of time intervals for which each label corresponding to an EV-nondominated path from a given origin has the least expected time is taken over all origins, all nondominated paths at each origin, and all 30 destinations.

4.3 Experimental Results

The results of these experiments are summarized in Tables I through IX. The number of nodes in the networks is indicated by the heading Nodes, the number of time intervals by TI, and the number of elements in the PMFs by Prob. The results are averaged for the networks that can be specified by the same (n, t, p)-triple (equivalent Nodes, TI, and Prob). In all of the tests, the SE list of each algorithm is implemented as a deque list (see Pape, 1974 for additional detail). These tests are not intended to test the performance of the procedures under a variety of SE list structures.

4.3.1 Run Times

The actual average run times for the EV and ELB procedures are given in Table I. To characterize the performance of the EV and ELB algorithms for this class of networks, the natural log of the run time in cpu milliseconds is regressed against the natural log of the number of nodes, number of time intervals,

[9]User cpu time refers to the cpu run time used only by the individual program; and therefore, accounts for time used by other programs simultaneously running on the same server.

TABLE I

Run times in cpu seconds for EV and ELB algorithms.

Nodes	Prob.	TI = 10	TI = 30	TI = 60	TI = 90
EV Algorithm					
50	5	0.01	0.05	0.11	0.19
	10	0.02	0.06	0.14	0.22
	20	0.02	0.10	0.23	0.37
100	5	0.02	0.10	0.25	0.44
	10	0.03	0.18	0.30	0.50
	20	0.05	0.20	0.50	0.82
500	5	0.14	0.64	2.08	3.27
	10	0.17	0.92	2.28	3.70
	20	0.30	1.47	3.54	5.54
1000	5	0.27	1.32	2.80	6.54
	10	0.36	1.76	4.36	6.94
	20	0.58	2.71	6.64	10.50
ELB Algorithm					
50	5	0.004	0.017	0.022	0.043
	10	0.007	0.023	0.048	0.080
	20	0.012	0.041	0.093	0.155
100	5	0.007	0.028	0.062	0.109
	10	0.014	0.069	0.108	0.174
	20	0.025	0.086	0.203	0.344
500	5	0.052	0.196	0.579	0.993
	10	0.090	0.393	0.985	1.460
	20	0.180	0.724	1.604	2.594
1000	5	0.102	0.428	1.123	1.865
	10	0.176	0.775	1.775	2.787
	20	0.359	1.050	3.096	4.830

TABLE II

Average and maximum number of EV-nondominated paths at each node for EV algorithm (rounded to nearest whole number)

Nodes	Prob.	TI = 10	TI = 30	TI = 60	TI = 90
Average at Each Node					
50	5	2	2	3	3
	10	2	2	2	2
	20	2	2	2	2
100	5	2	3	3	3
	10	2	2	2	2
	20	1	2	2	2
500	5	2	3	3	4
	10	1	2	2	2
	20	1	2	2	2
1000	5	2	3	3	3
	10	1	2	2	2
	20	1	2	2	2
Maximum at Any Node					
50	5	5	10	10	11
	10	5	9	9	9
	20	4	8	9	9
100	5	6	10	12	12
	10	5	9	10	10
	20	5	9	10	11
500	5	7	13	17	18
	10	6	11	14	14
	20	5	10	14	14
1000	5	7	13	17	18
	10	6	10	14	15
	20	5	9	14	15

and number of elements in the PMFs, resulting in the following equations:

$$T_{\mathrm{EV}} = (0.0024)(n^{1.1})(t^{1.4})(p^{0.5}), \tag{4}$$

$$T_{\mathrm{ELB}} = (0.00059)(n^{1.2})(t^{1.2})(p^{0.8}), \tag{5}$$

where T_{EV} and T_{ELB} are the average of the 30 pertinent run times, corresponding to the 30 randomly selected destination nodes, for an (n, t, p)-triple.

As suggested by Table I and the estimated coefficients of regression Eqs. 4 and 5, the run times of both the EV and ELB algorithms grow nearly linearly with the number of nodes, somewhat worse than linearly with increasing number of time intervals, and better than linearly with increasing number of elements in the PMFs. The EV algorithm takes approximately 2–3 times longer than the ELB algorithm to terminate.

4.3.2 The Number of EV-Nondominated Paths

The average and maximum numbers of EV-nondominated paths are given in Table II to assess the second hypothesis that the number of paths with the least expected time for at least one time interval is small. As shown, the average number of EV-nondominated paths from each origin node is between 1 and 4, and most often 2. This does not appear to be affected by the size of the network and grows only slightly with the number of time intervals. The maximum number of EV-nondominated paths from an origin is reasonably low for the EV algorithm and appears to be nearly unaffected by the number of elements in the PMFs, with the exception that the number is consistently highest for the lowest value, Prob. = 5.

4.3.3 Label Comparisons

The average number of label comparisons for the EV algorithm are given in Table III. To contrast this with the ELB procedure, which only requires one label vector for each node, the results of similar tests on the ELB procedure are also given. These results are intended to assess the third hypothesis that the number of labels required for determining these paths does not grow exponentially with the size of the network.

Table III shows that the number of label comparisons (comparisons between two labels over the peak period) increases linearly with increasing number of nodes, and decreases slightly with increasing number of elements in the PMFs for both the EV and ELB algorithms. There is nearly no increase in the number of label comparisons of the ELB algorithm

TABLE III

Average number of label comparisons for the EV and ELB algorithms

Nodes	Prob.	TI = 10	TI = 30	TI = 60	TI = 90
EV Algorithm					
50	5	444	673	792	936
	10	364	510	528	549
	20	403	483	503	507
100	5	836	1382	1657	1920
	10	679	974	1038	1034
	20	369	909	975	976
500	5	4245	7530	9866	11268
	10	3353	5080	5702	5739
	20	3725	4489	5050	5101
1000	5	7488	12892	17024	19551
	10	6004	8680	10049	10270
	20	5699	7506	8805	8879
ELB Algorithm					
50	5	230	236	252	266
	10	212	213	214	214
	20	210	210	210	210
100	5	450	476	499	523
	10	412	415	419	420
	20	408	409	409	409
500	5	2362	2483	2651	2774
	10	2158	2170	2175	2190
	20	2090	2085	2087	2084
1000	5	4522	4761	5014	5246
	10	4108	4131	4176	4183
	20	3957	3960	3971	3958

with increasing number of time intervals; however, there is significant increase in this factor with increasing number of time intervals for the EV algorithm. The increased number of comparisons of the EV algorithm, as compared to the ELB algorithm, is a consequence of the number of comparisons required of each new label that is constructed. That is, more than one label may be p-optimal, and, thus, the new label may be compared to a set of labels, as opposed to a single label in the ELB algorithm, over the time period. The growth in the number of comparisons with an increasing number of time intervals can be explained by the fact that an increasing number of time intervals leads to a higher likelihood of a path having the least expected time for at least one time interval. However, for the EV algorithm, there is no indication that an exponentially growing number of label comparisons is made with increasing network size, suggesting that the number of labels required to obtain the final least expected time paths does not grow exponentially with the network size.

4.3.4 Tightness of the ELB

Three measures are used to assess the fourth hypothesis (the lower bounds on the expected time determined by the ELB procedure are close to the expected times of the least expected time paths). The results of the related tests are presented next. Additional testing is conducted to compare these results for the networks of average in- and out-degree of 4 to networks with higher average in- and out-degree.

TABLE IV

Percent relative difference of the results from the ELB algorithm (to 2 significant figures)

Diff.	Prob.	TI = 30	TI = 90
% = 0%	5	82	79
% ⩽ 1%	5	91	88
% ⩽ 2%	5	96	94
% ⩽ 3%	5	98	97
% ⩽ 4%	5	100	99
% ⩽ 5%	5	100	100
% = 0%	20	86	82
% ⩽ 1%	20	96	94
% ⩽ 2%	20	100	99
% ⩽ 3%	20	100	100

Percent Relative Difference: Table IV gives the results of the tests to determine the percent relative difference of the lower bound computed by the ELB procedure and the true expected times of the least expected time paths. The first column indicates the level of the percentage of solutions that are less than the indicated percent difference. For example, the second row shows that, for 30 time intervals and 5 elements in the PMFs, 91 of 100 solutions of the ELB algorithm have a relative difference from the time of the least expected time paths of 0.01. The results given in this table are taken from the average of the three 500-node networks with average degree approximately 4, as described previously. Only 30 and 90 time intervals and 5 and 20 elements in the PMFs are considered. For approximately 80% of the results, the lower bound is identical to the actual time on the least expected time path for the given time interval. Approximately 90% of the results are within 1% relative difference of the corresponding least expected time. Although this cannot be seen from Table IV because the table gives the results for two significant figures, there are a few outlying results for which the worst lower bound has a greater than 5% relative difference. More extensive testing is required to determine the effects of the size of the network.

Maximum Percent Relative Difference: Table V shows the maximum percent relative difference from any node to any of the 30 destinations for the given time intervals and number of elements in the PMFs. The results given in this table are taken from the average of the three 500-node networks with

TABLE V

Maximum percent relative difference

Prob.	TI = 30	TI = 90
5	12.5	12.7
20	4.3	4.6

average degree approximately 4, as described previously.

Average Percent Relative Difference: Similarly, Table VI shows the average percent relative difference over all origins to any of the 30 destinations considered for the given time intervals and number of elements in the PMFs. The results given in this table are taken from the average of the three 500-node networks with average degree approximately 4, as described previously. All values are less than 1%.

4.3.5 Percent Relative Difference Measures for Denser Networks

From Proposition 5, it is apparent that the larger the number of possible paths from a node, the more likely a label from an upstream node will be composed of labels corresponding to a mixture of times from several paths. It seems likely that the larger the average degree of a network, the further the lower bound determined by the ELB algorithm will be from the expected time on the true least expected time path. Two additional 500-node networks with an average in- and out-degree of approximately 10 (with a range of 2–19) are randomly generated to verify this assertion. Only 30 and 90 time intervals and 5 and 20 elements in the PMFs are considered. The average results from these two 500-node networks are summarized in Tables VII and VIII.

Comparing the results of the 4 degree networks in Tables IV–VI with the results of the 10 degree networks in Tables VII and VIII indicates that the relative difference in the solutions increases with increasing average degree. However, even for an average degree of 10, more than 80% of the solutions are within 1% of the true least expected time. Several more networks of varying average degree would need to be tested to generalize these results. Because the principal focus of this work is on transportation systems, further testing is beyond the scope of this paper.

TABLE VI

Average percent relative difference

Prob.	TI = 30	TI = 90
5	0.24	0.31
20	0.10	0.15

TABLE VII

Percent relative difference of the results from the ELB algorithm for the network with degree 10 (to 2 significant figures)

Diff.	Prob.	TI = 30	TI = 90
% = 0%	5	79	77
% ⩽ 1%	5	86	84
% ⩽ 2%	5	90	89
% ⩽ 3%	5	94	93
% ⩽ 4%	5	96	95
% ⩽ 5%	5	98	97
% ⩽ 6%	5	99	99
% ⩽ 7%	5	100	99
% ⩽ 8%	5	100	100
% = 0%	20	83	80
% ⩽ 1%	20	91	88
% ⩽ 2%	20	97	96
% ⩽ 3%	20	99	99
% ⩽ 4%	20	100	100

4.3.6 Robustness of the EV-Nondominated Paths

The EV algorithm terminates with all the EV-nondominated paths. Each EV-nondominated path may be the least expected time path for only a subset of time intervals. To test the robustness of the solutions (Hypothesis 5), the average number of time intervals for which each EV-nondominated path is the least expected time path for the test networks is obtained. The results of these tests, rounded to the nearest whole number, are given in Table IX.

In general, each EV-nondominated path has the least expected time for approximately one-half of the time intervals. This is consistent with the results of Table II where, on average, two (between one and four) least expected time paths exist from each origin node. Such solutions are considered robust.

5. CONCLUSIONS AND DISCUSSION

TWO ALGORITHMS FOR solving the LET path problem in STV networks were presented. The EV algorithm determines the a priori least expected time paths with the associated expected time from all nodes to a given destination for each departure time interval in the period of interest. Although worst-case com-

TABLE VIII

Maximum and average percent relative difference for networks with degree 10

	Maximum		Average	
Prob.	TI = 30	TI = 90	TI = 30	TI = 90
5	14.9	17.5	0.48	0.55
20	7.1	6.7	0.27	0.60

TABLE IX

Average number of time intervals for which each EV-nondominated path has the least expected time for the EV algorithm

Nodes	Prob.	TI = 10	TI = 30	TI = 60	TI = 90
50	5	6	14	25	35
	10	7	16	30	45
	20	7	16	31	46
100	5	6	13	24	34
	10	7	16	30	44
	20	7	17	31	46
500	5	6	13	22	31
	10	7	16	29	42
	20	7	17	31	45
1000	5	6	14	23	33
	10	7	17	30	44
	20	8	18	32	47

putational complexity is nonpolynomial, for networks with an average in- and out-degree each of 4 from the nodes, the average actual computational effort was estimated to be $\sim T_{\text{avg}}(I^{1.4}v^{1.1}P^{0.5})$, where I is the number of time intervals in the peak period, v is the cardinality of nodes in G, and P is the number of elements in the PMFs. For very large networks, or dense networks, the number of paths that may be examined can grow quite large, and, thus, this algorithm may perform rather poorly. Hence, an efficient method is required. The ELB algorithm is an efficient algorithm that determines a lower bound on the expected times of the LET paths. Although this algorithm does not give any path information, in every case examined, including 500-node networks with an average in- and out-degrees each of 10, at least 73% of the solutions were identical to the actual least expected path times. Thus, by saving path information in the course of the algorithm, the paths of at least these labels would be correctly identified. The ELB algorithm also provides an exact solution to the problem of identifying the LET path where the driver is permitted to react to revealed travel times on traveled arcs en route (i.e., in a time-adaptive route choice framework, also known as a routing problem with recourse).

The average performance of each algorithm is clearly better than predicted by worst-case computational complexity analyses. Several factors may contribute to this average performance. In the case of the ELB algorithm, the networks used in the tests are sparse, whereas, in the worst-case analyses, the networks are assumed to be complete. From the test results, it appears that the times found by the ELB algorithm are likely to be closer to the true least expected path times in sparse networks, because fewer paths contribute to the construction of these bounds in sparser networks.

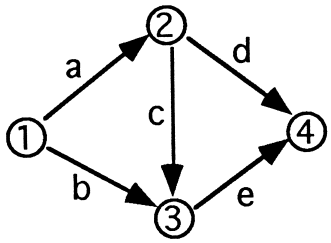

Fig. 1. An example network.

The EV algorithm also performs better than worst-case analysis predicts. That is, the number of paths that are nondominated in pairwise comparisons does not grow exponentially with the size of the network for the class of networks tested.

In this work, the PMFs were generated such that each arc has the same likelihood of having the same PMF. If the arc PMFs are similar, it is more likely that there will be several EV-nondominated paths at each node. The run time of the EV algorithm will be reduced if only a few outstanding paths or subpaths exist, as is likely to occur in transportation networks. Furthermore, a path that is best in one time interval is likely to continue to be best in many consecutive time intervals. This correlation is not considered here. Further investigation of this aspect may lead to valuable insights into the performance of these algorithms and into the structure of the problem itself.

In addition to the tests described in Section 4, both algorithms were run on a network with 9000 nodes and over 36,000 arcs. The ELB algorithm was also run on a network of 15,000 nodes, 61,386 arcs, 30 time intervals, and 5 elements in the PMFs, with average run time over 30 randomly generated destination nodes of 12.5 user cpu seconds.

APPENDIX A

BOTH THE EV AND ELB algorithms take advantage of the fact that the expected value of a path can be calculated from the distribution of a single arc at a given departure time and the expected times over all time intervals of the remaining subpath. For example, in Figure 1, the expected time on path 1–2–3–4 at departure time interval 0 can be determined from the PMF of arc a at departure time 0 and the expected travel times of subpath 2–3–4 at all the possible arrival times at node 2. The travel time PMFs for the example network in Figure 1 are given in Table X where the travel time (probability) are given for each arc at the relevant time intervals. Travel times can be in any unit of time, assume in minutes.

TABLE X

Travel times and associated probabilities for the example network shown in Figure 1

Arc a	Arc b	Arc c		Arc d		Arc e			
$t = 0$	$t = 0$	$t = 2$	$t = 3$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$
2 (0.5)	5 (0.4)	4 (0.8)	1 (0.3)	3 (0.8)	6 (0.4)	4 (0.2)	5 (0.3)	1 (0.9)	3 (0.3)
3 (0.5)	7 (0.6)	5 (0.2)	3 (0.7)	7 (0.2)	7 (0.6)	6 (0.8)	8 (0.7)	2 (0.1)	4 (0.7)

The expected travel times can be calculated as follows:

Expected time on subpath 2–3–4 at $t = 2$

$$(4 + 1.1)(0.8) + (5 + 3.7)(0.2) = 5.82 \text{ minutes.}$$

Expected time on subpath 2–3–4 at $t = 3$

$$(1 + 5.6)(0.3) + (3 + 8.3)(0.7) = 9.89 \text{ minutes.}$$

Expected Time on path 1–2–3–4 at $t = 0$

$$(2 + 5.82)(0.5) + (3 + 9.89)(0.5) = 10.355 \text{ minutes.}$$

APPENDIX B: EXAMPLE PROBLEM

FOR THE EXAMPLE in Figure 1 and Table X of Appendix A, determine the lower bound on the least expected time for any path from node 1 to node 4 at departure time 0. The associated $\pi_i(t)$ for the time-adaptive route choice problem are also indicated.

0. Initialization
   $\lambda_i(t) = \infty$ and $\pi_i(t) = \infty$
   $\forall i \in \{1, 2, 3\}$, $t \in \mathcal{S} = \{0, 1, \ldots, 7\}$.
   $\lambda_4(t) = 0$ and $\pi_4(t) = \varnothing$ $\forall t \in \mathcal{S}$.
   SE = {4}.
1. Scan node 4.
2. For each $i \in \Gamma^{-1}(4) = \{2, 3\}$, determine the lower bound on the expected time to node 4.
   $\eta_2(2) = (3 + 0)(0.8) + (7 + 0)(0.2) = 3.8 < \infty$,
   $\lambda_2(2) = \mathbf{3.8}$, $\pi_2(2) = d$.
   $\eta_2(3) = (6 + 0)(0.4) + (7 + 0)(0.6) = 6.6 < \infty$,
   $\lambda_2(3) = \mathbf{6.6}$, $\pi_2(3) = d$.
   SE = {2}
   Similar calculations for node 3 lead to:
   $\lambda_3(4) = \mathbf{5.6}$, $\pi_3(4) = e$.
   $\lambda_3(5) = \mathbf{7.1}$, $\pi_3(5) = e$.
   $\lambda_3(6) = \mathbf{1.1}$, $\pi_3(6) = e$.
   $\lambda_3(7) = \mathbf{3.7}$, $\pi_3(7) = e$.
   SE = {2, 3}
   1. Scan node 2—for all $i \in \Gamma^{-1}(2) = \{1\}$
   2. $\eta_1(0) = (2 + 3.8)(0.5) + (3 + 6.6)(0.5) = 7.7 < \infty$,
      $\lambda_1(0) = \mathbf{7.7}$, $\pi_1(0) = a$.
   SE = {3}
   1. Scan node 3—for all $i \in \Gamma^{-1}(3) = \{1, 2\}$
   2. $\eta_1(0) = (5 + 7.1)(0.4) + (7 + 3.7)(0.6) = 11.26 > 7.7$
      $\eta_2(2) = (4 + 1.1)(0.8) + (5 + 3.7)(0.2) = 5.82 > 3.8$
      $\eta_2(3) = (1 + 5.6)(0.3) + (3 + 1.1)(0.7) = 4.85 < 6.6$,
      $\lambda_2(3) = \mathbf{4.85}$, $\pi_2(3) = c$.
   SE = {2}
      $\eta_1(0) = (5 + 7.1)(0.4) + (7 + 3.7)(0.6) = 11.26 > 7.7$
   1. Scan node 2—for all $i \in \Gamma^{-1}(2) = \{1\}$
   2. $\eta_1(0) = (2 + 3.8)(0.5) + (3 + 4.85)(0.5) = 6.825 < 7.7$,
      $\lambda_1(0) = \mathbf{6.825}$, $\pi_1(0) = a$.
   SE = { }
3. Stop

Note that in this example, calculations for $\lambda_1(t)$ and $\pi_1(t)$ for $t = 0$ only are shown. The algorithm actually calculates, $\pi_i(t)$ $\forall i \in \mathcal{V}$ and $t \in \mathcal{S}$.

A Priori LET Paths

If the subpaths of the a priori LET path are the LET paths for all time intervals, then this algorithm will determine the LET of a single path, which can be identified through the use of path pointers. This example is constructed such that, for each possible arrival time at node 2, a different path would have the least expected travel time. Thus, more than one path contributes to the node label at node 1. In the example problem, path 2–4 is best for time interval 2 and path 2–3–4 is best for time interval 3. Therefore, the label from node 1 through node 2 uses the expected times from both paths 2–4 and 2–3–4 and the label will have a lower value than the expected time on either path 1–2–4 or 1–2–3–4. The least expected travel time for any path in the network is 7.7 minutes and 6.825 minutes is a lower bound on this duration.

LET Paths for Time-Adaptive Route Choice

The resulting hyperpaths for this problem formulation are given in the form of a tree as shown in Figure 2. For a given arc corresponding to the next arc on the path, at the given departure time t from the arc's origin node, the EV for the path is given. For example, if the travel time on arc (1, 2) is 3

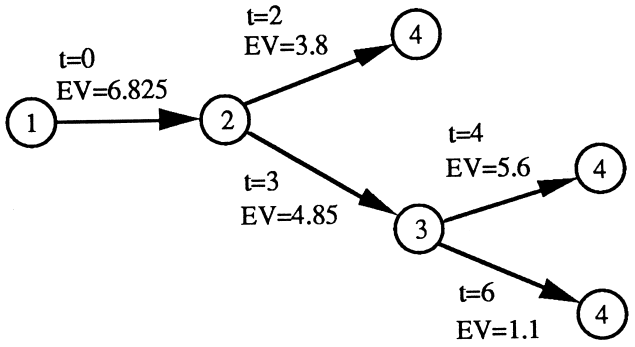

Fig. 2. Resulting hyperpaths as shown through conditional tree structure.

minutes, then the arrival time at node 2 is $t = 3$. The driver will choose arc (2, 3) next and the remainder of the path to node 4 will have an expected time of 4.85 minutes.

ACKNOWLEDGMENTS

THE AUTHORS ARE grateful to Dr. Athanasios Ziliaskopoulos, presently at Northwestern University, for many useful discussions in the general area of multidimensional optimum path algorithm implementation. The authors would like to thank the anonymous referees for their valuable comments. We are especially grateful to one of the referees who sharpened our understanding of the usefulness of the ELB algorithm. Partial funding for the work reported herein came from a research contract through the Amarillo National Research Center for Plutonium, and from support through the Southwest (Region 6) University Transportation Center.

REFERENCES

R. AHUJA, "Network Flows," in *Handbooks in Operations Research and Management Science: Volume 1, Optimization*, G. Nemhauser, A. Kan and M. Todd (eds.), North-Holland, New York, 1989. 261–263.

R. AHUJA, T. MAGNANTI, AND J. ORLIN, *Network Flows*, Chapter 5, Prentice-Hall, New Jersey, 1993. 140–141.

R. BELLMAN, "On a Routing Problem," *Quart. Appl. Math.* **16,** 87–90 (1958).

J. BIRGE AND F. LOUVEAUX, *Introduction to Stochastic Programming*, Springer-Verlag, New York, 1997.

J. BRUMBAUGH-SMITH AND D. SHIER, "An Empirical Investigation of Some Bicriterion Shortest Path Algorithms," *Eur. J. Oper. Res.* **43,** 216–224 (1989).

A. CHARNES, W. COOPER, AND G. THOMPSON, "Critical Path Analyses via Chance Constrained and Stochastic Programming," *Opns. Res.* **12,** 460–470 (1964).

G. COREA AND V. KULKARNI, "Shortest Paths in Stochastic Networks with ARC Lengths having Discrete Distributions," *Networks* **23,** 175–183 (1993).

A. EIGER, P. MIRCHANDANI, AND H. SOROUSH, "Path Preferences and Optimal Paths in Probabilistic Networks," *Transp. Sci.* **19,** 75–84 (1985).

H. FRANK, "Shortest Paths in Probabilistic Graphs," *Opns. Res.* **17,** 583–599 (1969).

G. GALLO AND S. PALLOTTINO, "Shortest Path Methods: A Unifying Approach," *Math. Prog. Study* **26,** 38–64, North-Holland, Amsterdam, 1986.

F. GLOVER, D. KLINGMAN, N. PHILLIPS, AND R. SCHNEIDER, "New Polynomial Shortest Path Algorithms and their Computational Attributes," *Management Sci.* **31,** 1106–1128 (1985).

R. HALL, "The Fastest Path through a Network with Random Time-Dependent Travel Times," *Transp. Sci.* **20,** 182–188 (1986).

P. HANSEN, "Bicriterion Path Problem," in *Multiple Criteria Decision Making: Theory and Applications*, Lecture Notes in Economics and Mathematical Systems 177, 109–127, Springer-Verlag, Berlin, 1980.

D. KAUFMAN AND R. SMITH, "Fastest Paths in Time-Dependent Networks for Intelligent Vehicle Highway Systems Applications," *IVHS J.* **1,** 1–11 (1993).

V. KULKARNI, "Shortest Paths in Networks with Exponentially Distributed Arc Lengths," *Networks* **16,** 255–274 (1986).

R. LOUI, "Optimal Paths in Graphs with Stochastic or Multidimensional Weights," *Comm. ACM* **26,** 670–676 (1983).

A. MADANSKY, "Inequalities for Stochastic Linear Programming Problems," *Management Sci.* **6,** 197–204 (1960).

D. MALCOLM, J. ROSEBOOM, AND C. CLARK, "Application of Technique for Research and Development Program Evaluation," *Opns. Res.* **7,** 646–669 (1959).

E. MILLER-HOOKS, Optimal Routing in Time-Varying, Stochastic Networks: Algorithms and Implementation, Ph.D. dissertation, Department of Civil Engineering, The University of Texas at Austin, 1997.

E. MILLER-HOOKS AND H. MAHMASSANI, "Least Possible Time Paths in Stochastic, Time-Varying Networks," *Comput. Opns. Res.* **25,** 1107–1125 (1998).

P. MIRCHANDANI AND H. SOROUSH, "Optimal Paths in Probabilistic Networks: A Case with Temporary Preferences," *Comput. Opns. Res.* **12,** 365–381 (1985).

I. MURTHY AND S. SARKAR, "A Relaxation-Based Pruning Technique for a Class of Stochastic Shortest Path Problems," *Transp. Sci.* **30,** 220–236 (1996).

S. NGUYEN AND S. PALLOTTINO, "Hyperpaths and Shortest Hyperpaths," in *Combinatorial Optimization*, Lecture Notes in Mathematics 1403, 258–271, Springer-Verlag, Berlin, 1986.

A. ORDA AND R. ROM, "Minimum Weight Paths in Time-Dependent Networks," *Networks* **21,** 295–319 (1991).

S. PALLOTTINO, "Shortest-Path Methods: Complexity, Interrelations and New Propositions," *Networks* **14,** 257–267 (1984).

U. PAPE, "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem," *Math. Program.* **7,** 212–222 (1974).

H. PSARAFTIS AND J. TSITSIKLIS, "Dynamic Shortest Paths

in Acyclic Networks with Markovian Arc Costs," *Opns. Res.* **41,** 91–101 (1993).

C. SIGAL, A. PRITSKER, AND J. SOLBERG, "The Stochastic Shortest Route Problem," *Opns. Res.* **28,** 1122–1129 (1980).

R. VAN SLYKE, "Monte Carlo Methods and the PERT Problem," Rand Research Memorandum RM-3367-PR, 1963.

A. ZILIASKOPOULOS, "Design and Implementation of Some K Shortest Path Algorithms with Application to Intelligent Vehicle Highway Systems," Master's Report, Department of Civil Engineering, The University of Texas at Austin, 1992.

A. ZILIASKOPOULOS AND H. MAHMASSANI, "Design and Implementation of a Shortest Path Algorithm with Time-Dependent Arc Costs," *Proc. 5th Advanced Technology Conference*, 1179–1194, U.S. Postal Service, Washington, D.C., 1992.

A. ZILIASKOPOULOS AND H. MAHMASSANI, "Time-Dependent, Shortest-Path Algorithm for Real-Time Intelligent Vehicle Highway System Applications," *Transp. Res. Rec.* **1408,** 94–100 (1993).

A. ZILIASKOPOULOS, "Optimum Path Algorithms on Multidimensional Networks: Analysis and Design, Implementation and Computational Experience," Ph.D. dissertation, Department of Civil Engineering, The University of Texas at Austin, 1994.

(Received: December 1997; revisions received: October 1998; Accepted: November 1998)