# On the Online Shortest Path Problem with Limited Arc Cost Dependencies

**S. Travis Waller**
*Department of Civil and Environmental Engineering, University of Illinois at Urbana–Champaign, Urbana, Illinois 61801*

**Athanasios K. Ziliaskopoulos**
*Department of Civil Engineering, Northwestern University, Evanston, Illinois 60208*

**This paper is concerned with the stochastic shortest path problem with recourse when limited forms of spatial and temporal arc cost dependencies are accounted for. Recourse is defined as the opportunity for a decision maker to reevaluate his or her remaining path when en-route information is available. Formulations with recourse typically provide opportunities for corrective actions when information becomes available; information here is modeled as arc cost dependencies, defined as spatial and temporal. System properties are stated and proved and solution algorithms are developed for limited cases of spatial and temporal arc cost dependencies. The numerical results verify some of the theoretical insights and demonstrate the applicability of the introduced algorithms.** © 2002 Wiley Periodicals, Inc.

**Keywords:** shortest paths; stochastic optimization; online algorithms

## 1. INTRODUCTION

The deterministic Shortest Path Problem (SPP) has been extensively studied in the literature and various efficient algorithms have been presented. Recent developments in the field of stochastic optimization have motivated revisiting many of the traditional network problems with relaxed assumptions on the determinism of various parameters. These stochastic network problems can be grouped in two general classes: The first class aims to find one *a priori* solution that minimizes the expected cost, while the second computes an online solution that allows decisions to be made at various stages (also known as a recourse problem). Much of the research on *a priori* stochastic SPPs has dealt with dynamic variations of the problem. An efficient heuristic algorithm for this problem was given in Fu and Rilett [4], while optimal, nonpolynomial ones were presented in Hall [5] as well as in Miller-Hooks and Mahmassani [6].

In the context of the SPP, recourse can be viewed as the opportunity for a decision maker to reevaluate his or her remaining path at each node based on knowledge obtained en route. For example, a driver may enter a highly congested road segment, and after traversing it, he or she may reassess the successor segments he or she should follow, knowing that the predecessor one was congested. This illustrates spatial dependence, where information is obtained for successor arcs while traversing the network. Temporal dependence suggests a dependency between time periods for the same arc; this can be seen in systems with real-time monitoring. If a vehicle were informed on the value of arc costs when entering a network, temporal dependence would take into account the change of the expected cost of the network over time.

Croucher [3] introduced an approach that accounts for recourse actions with a heuristic algorithm on a restricted model. Andreatta and Romeo [2] presented properties of recourse models on networks with a general dependency and developed a polynomial algorithm for independently distributed arc costs and deterministic recourse—if a blocked arc is encountered, a recourse path consisting only of deterministic arcs is used. Polychronopoulos and Tsitsiklis [7] proposed two models with recourse. The first model assumes general spatial dependence among the arcs captured in the form of realizations. These realizations represent all possible network instantiations, and, therefore, the knowledge obtained from traversing along the network reduces the possible realizations. An exponential algorithm—with respect to the number of realizations—was constructed for this model. The second model proposed is a special case of the first one where the arc costs are independently dis-

Received January 2001; accepted August 2002
*Correspondence to:* A. K. Ziliaskopoulos;
e-mail: a-z@northwestern.edu
Contract grant sponsors: NSF; contract grant number: 0830-350-C653; FHWA/NSF; contract grant number: 0830-350-C819; Northwestern University's Transportation Center; Louis Berger Junior Chair Professorship

tributed random variables. An exponential algorithm with respect to the number of arcs was constructed for such problems. An algorithm for the problem of recourse on time-varying networks was proposed in Miller-Hooks and Mahmassani [6]. While the algorithm determines an optimal least-expected cost solution with respect to the arrival time at a node, it assumes independence between both arc costs and time periods.

In this paper, the online SPP is addressed when various forms of local dependency are present. Properties of the solution, such as cycling, are explored and efficient algorithms are presented for special cases. The issue of spatial dependency, as examined in Polychronopoulos and Tsitsiklis [7], is addressed. However, instead of using all possible network instantiations as states to account for general dependency, we assume limited interarc dependencies only. This results in an approach where node realizations (states for emanating arcs) depend only on conditions of surrounding arcs.

SPPs on networks with such a dependency are called *Spatially Dependent Online Shortest Path* (SD-OSP) problems; we introduce an algorithm for this class of problems assuming one-step spatial dependence. This implies that if information from the predecessor arc is given, no further spatial information has an impact on the expected current arc cost. An assumption made for the complexity proof of every algorithm is that arc costs are strictly greater than zero. It will be further assumed, throughout this work, that each visit to an arc results in another random trial and that waiting will not be allowed at nodes (although cycling is acceptable). In the spatial variants of this problem, it is assumed that interarc dependencies exist for adjacent links only.

Furthermore, an implicit assumption within virtually all time-invariant stochastic shortest path work is that the cost of an arc $(a,b)$ is learned once node $a$ is reached. This assumption will be made within the limited temporal variants of the problem only and will be assumed in the property analysis only when explicitly stated. While this assumption may not appear very restrictive, it can be interpreted as a type of temporal dependence and could substantially impact the solution of the recourse problem. First, it presumes that some means of gathering downstream information (e.g., a real-time monitoring system) is present. Second, it assumes that arc $(a,b)$ can be traversed in a reasonably short time, so that the cost on the arc $(a,b)$ will not change during its traversal. SPPs on networks with such a dependency are called *Temporally Dependent Online Shortest Path* (TD-OSP) problems. An algorithm for this class of problems is introduced as well as for problems that combine spatial and temporal dependence—*Temporally Spatially Dependent Online Shortest Path* (TSD-OSP) problems. Each of the introduced algorithms is shown to perform in polynomial time in the absence of cycling, while probabilistic cost bounds are developed when cycling occurs. The arc cost dependency within the models can be interpreted as a convenient means of representing network information. Therefore, developing algorithms and properties for the presented problems has value even though these models may not hold for many practical applications. Note that previous efforts on developing approaches that account for dependency have been limited to algorithms with a computational time exponential in the size of the problem.

In Section 2, basic properties of the examined OSP problems are investigated. Efficient algorithms for these problems are introduced in Section 3; algorithm characteristics are explored and their computational complexities are derived. Computational experiments performed on a limited set of networks in Section 4 support the theoretical insights and computational bounds derived in Section 3. Finally, Section 5 concludes this research and provides pointers for further research.

## 2. PROBLEM ANALYSIS

Let $G = (N,A,P)$ be a probabilistic directed network with a node set $N$ ($|N| = n$), an arc set $A$ ($|A| = m$), and a set of probability matrices $P$ that consists of the discrete probability distributions for each arc cost. A matrix $P$ will specify, at a minimum, the probability for each possible state on every arc where the rows are indexed by the current arc state, and if conditional probabilities are employed, the columns are indexed by the previous arc state. Denote by $\Gamma^{-1}(i)$ the set of predecessor nodes of node $i \in N$, and $\Gamma(i)$, the set of successor nodes to $i$. In addition, let $S(i,j)$ denote the set of all possible states for arc $(i,j)$; arc cost $c_s^{i,j}$ refers to the cost of arc $(i,j)$ at state $s \in S(i,j)$. Finally, let $t \in N$ be a destination or the terminal node. The expected cost from node $j$ to the destination node $t$ is denoted by $E[j]$, while $E[j|i,s]$, denotes the expected cost from $j$ to the destination node $t$, given that $(i,j) \in A$ is at state $s$, $s \in S(i,j)$. Finally, it is assumed that arc costs do not become deterministically known after they are visited, that is, each visit to an arc is an independent stochastic trial.

### *Online Shortest Path Properties*

In contrast to *a priori* SPPs where the goal is to find a single path with the least expected cost, the goal of the algorithms introduced in this paper is to find a solution with the least expected cost. This solution consists of a collection of state-paths, each associated with a cost and probability for being followed. While the recourse solution does not necessarily require the explicit enumeration of all possible paths, we will consider such an enumeration so that the recourse properties can be demonstrated. As an example of the paths that might make up this solution, consider the network in Figure 1, where the assumption is made that the cost of an arc $(i,j)$ is learned when node $i$ is reached and repeated visits to this arc will result in independent stochastic trials.

A traveler of this network would cycle whenever he or she arrives at node $c$ and the cost on arc $(c,d)$ is 101. A solution to this problem would consist of a discrete proba-

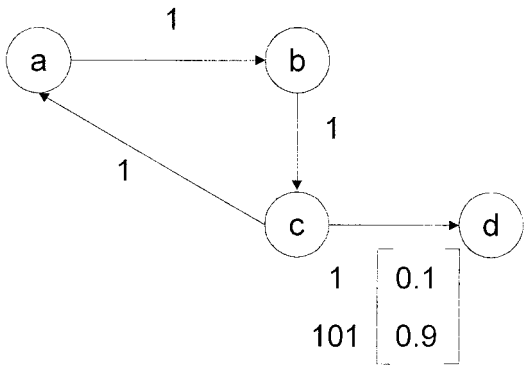


FIG. 1. Network in which the OSP may include cycles.

bility distribution describing the path to be followed and its cost. Let a path, described as a sequence of nodes, be associated with a cost and the corresponding probability. If only acyclic state-paths were considered, an optimal solution would be

$$\begin{array}{lll} a\text{–}b\text{–}c\text{–}d & \text{Cost} = 3 & \Pr = 0.1 \\ a\text{–}b\text{–}c\text{–}d & \text{Cost} = 103 & \Pr = 0.9. \end{array}$$

This solution could be interpreted as simply taking path *a–b–c–d* regardless of any information learned en route and would yield an expected cost of 93. If one cycle alone were allowed, an optimal solution would be

$$\begin{array}{lll} a\text{–}b\text{–}c\text{–}d & \text{Cost} = 3 & \Pr = 0.1 \\ a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}d & \text{Cost} = 6 & \Pr = 0.09 \\ a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}d & \text{Cost} = 106 & \Pr = 0.81. \end{array}$$

This solution could be interpreted as taking subpath *a–b–c*, then *d* if a cost of 1 was found on *c–d*. If a cost of *c–d* were found to be 101, the subpath *a–b–c–d* would then be followed regardless of any additional information. This would give an expected cost of 86.7. If two cycles were allowed,

$$\begin{array}{lll} a\text{–}b\text{–}c\text{–}d & \text{Cost} = 3 & \Pr = 0.1 \\ a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}d & \text{Cost} = 6 & \Pr = 0.09 \\ a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}d & \text{Cost} = 9 & \Pr = 0.081 \\ a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}a\text{–}b\text{–}c\text{–}d & \text{Cost} = 109 & \Pr = 0.729. \end{array}$$

This would result in a cost of 81.03. This process may continue indefinitely, with suboptimal state-paths being replaced as additional cycles are accounted for. In this simple example, the expected cost would be

$$E[\text{cost}] = 0.1 \times \sum_{i=0}^{\infty} 3 \times (i+1) \times 0.9^i = 30.$$

**Property 1.** *An optimal solution may contain paths consisting of an arbitrarily large number of arcs.*

**Proof.** As in other recourse models [2], cycles might exist in an optimal solution. However, in this context, the same cycle may be repeated infinitely many times in an optimal solution (as demonstrated in the example of the network in Fig. 1), since visits to an arc consist of independent stochastic trials. ∎

**Lemma 1.** *The expected cost resulting from the recourse solution will be less than or equal to the expected cost of the a priori solution.*

**Proof.** This follows from the fact that any optimal *a priori* path is a feasible online solution. ∎

*A Probabilistic Bound on the Number of Cycles*

It was established above that optimal solutions may include cycles; next, properties will be stated and proved that establish a probabilistic bound on the occurrence of cycling in an online optimum path. To develop such bounds, the following assumption is made: All arc costs are positive with $\xi$ being a lower possible bound on the cost and $\Xi$, an upper possible one, that is, $p(c_s^{i,j} \geq \xi) = 1$ and $p(c_s^{i,j} \leq \Xi) = 1$ $\forall (i,j) \in A$ and $\forall s \in S(i,j)$; $p(.)$ denotes the probability function. The cycling will be captured by the number of arcs in the optimum path; this number is a random variable denoted by $\eta$ and its cumulative distribution by $F(.)$. While it is difficult to produce closed-form expressions for this measure, several properties could be developed that also provide insights on the properties of cycling.

**Property 2.** *$\xi\, E[\eta] \leq n\Xi$, although $\eta$ itself may be unbounded.*

**Proof.** This follows from the fact that the solution to any *a priori* time-invariant problem would result in a single acyclic path and have a total cost of less than or equal to $n\Xi$. Since the recourse solution must have an expected cost less than or equal to the *a priori* solution, which is bounded above by $n\Xi$, and each arc visit results in at least a cost of $\xi$, then the number of arcs a traveler expects to traverse times the minimum cost cannot exceed the maximum *a priori* solution cost. ∎

**Property 3.** *$1 - F(a) \leq n\Xi/(\xi\, a)$ for any integer $a \geq 1$.*

**Proof.** $E[\eta]$ is defined as

$$E[\eta] = \sum_{i=0}^{\infty} i \times p(i).$$

where $p(i)$ is the probability of traversing exactly $i$ arcs. To develop a bound on $E[\eta]$, we define $\varepsilon(a)$ as follows:

$$\varepsilon(a) = a \sum_{i=a+1}^{\infty} p(i) = a \times (1 - F(a))$$

for any integer $a \geq 1$.

Since

$$E[\eta] = \sum_{i=0}^{\infty} i \times p(i) \geq \sum_{i=0}^{a} i \times p(i) + \sum_{i=a+1}^{\infty} a \times p(i) \geq a \sum_{i=a+1}^{\infty} p(i) = \varepsilon(a),$$

Property 2 gives $a\,(1 - F(a)) \leq E[\eta] \leq n\Xi/\xi$ or $1 - F(a) \leq n\Xi/(\xi a)$, as required. ∎

Property 2 essentially provides a loose bound on the maximum expected number of arcs in an online shortest path. It is used in proving Property 3, which states a lower bound on the cumulative distribution for the probability that a certain number of arcs will be followed, if an optimal online strategy is employed. In other words, a user of the network would know *a priori* the maximum probability that he or she would traverse more than a specified number of arcs. The maximum number of arcs visited can also be used to establish a bound on the maximum probability for a specified number of cycles to be taken. Therefore, if a particular online solution accounted for only a finite number of cycles, these properties can give confidence levels and performance bounds on the solution.

### *One-step Spatial Dependence*

The spatial dependence model (SD-OSP) assumes one-step arc dependence, that is, given the cost of predecessor arcs, no further information is obtained through spatial dependence. Since a traveler can only traverse one arc at a time, each arc is dependent on only one predecessor arc at a time, and this is reflected in the conditional probability matrices. For each node $c \in N$, $a \in \Gamma^{-1}(c)$, and $d \in \Gamma(c)$, $P^{a,c,d}$ describes the conditional probability of the state of arc $(c,d)$ given the state of the upstream arc $(a,c)$. Let $P^{a,c,d}_{s,k}$ denote the conditional probability that arc $(c, d)$ is in state $k$, given that arc $(a, c)$ was traversed in state $s$.

Next, consider the example network in Figure 2. Arc $(c, e)$ has a multidimensional probability matrix conditioned on either the state of arc $(a, c)$ or $(b, c)$. Suppose that each arc of the network has two states: uncongested ($U$) and congested ($C$), where the cost of state $C$ is greater than that of state $U$. Suppose that the probability matrices for arc $(c, e)$ are as follows:

$$P^{a,c,e} = \begin{array}{c} \\ U \\ C \end{array}\!\!\begin{array}{c} \begin{array}{cc} U & C \end{array} \\ \begin{pmatrix} .7 & .3 \\ .4 & .6 \end{pmatrix} \end{array} \qquad P^{b,c,e} = \begin{array}{c} \\ U \\ C \end{array}\!\!\begin{array}{c} \begin{array}{cc} U & C \end{array} \\ \begin{pmatrix} .5 & .5 \\ .5 & .5 \end{pmatrix} \end{array}.$$

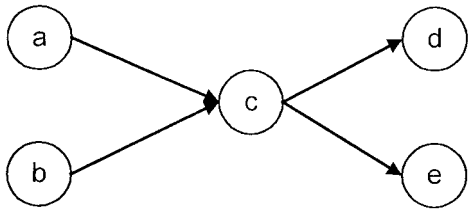


FIG. 2. Example network to demonstrate one-step spatial dependence.

In $P^{a,c,e}$, if arc $(a, c)$ is uncongested, then there is a 0.7 probability that arc $(c,e)$ is uncongested and, therefore, a 0.3 probability that arc $(c,e)$ is congested.

### *Limited Temporal Dependence*

The Limited Temporal Dependence (TD-OSP) model assumes that the cost of arc $(a,b)$ is known once node $a$ is reached. While this is the underlying assumption in many of the existing approaches on stochastic shortest path with recourse, it is considered separately here, so that the direct impact of this form of dependency can be better understood. This type of information is referred to here as temporal dependency, since it can be interpreted as yielding information on short-term future arc costs based on current observations for the same arc.

This limited temporal dependence can be seen as a first step toward a more general temporal dependence once this model is combined with the spatial approach. Furthermore, previous attempts at examining cases similar to even this limited temporal dependency have produced computationally intractable algorithms. An important attribute of the proposed approach is that each visit to an arc consists of an independent trial; the approaches by Andreatta and Romeo [2] and Polychronopoulos and Tsitsiklis [7] assumed that once an arc has been visited its cost becomes deterministically known. This is a difference that allows this class of problems to be solved efficiently.

For each node $b \in N$ and $c \in \Gamma(b)$, $P^{b,c}_s$ describes the probability that arc $(b,c)$ is in state $s$ independent of other arc costs. Again, consider the network in Figure 2. Suppose that the probability distributions are

$$P^{b,c} = P^{c,d} = P^{c,c} = \begin{array}{c} U \\ C \end{array}\!\!\begin{bmatrix} .5 \\ .5 \end{bmatrix}.$$

When a traveler arrives at node $c$, he or she learns what the costs on $(c,d)$ and $(c,e)$ will be. Therefore, the traveler can at this time minimize the expected cost to the destination while deterministically knowing the cost of arcs $(c,d)$ and $(c,e)$. This type of information can be derived from the combination of the arc probability matrices while taking the minimum cost for each possible realization over all potential arc costs at this node alone:

$$E^c = \begin{matrix} .25 \\ .25 \\ .25 \\ .25 \end{matrix}\begin{bmatrix} \min(c_U^{c,d} + E[d], c_U^{c,e} + E[e]) \\ \min(c_U^{c,d} + E[d], c_C^{c,e} + E[e]) \\ \min(c_C^{c,d} + E[d], c_U^{c,e} + E[e]) \\ \min(c_C^{c,d} + E[d], c_C^{c,e} + E[e]) \end{bmatrix}. \quad (2.1)$$

This represents the fact that a distinct cost can be placed on a node for every possible state that might occur due to random costs from emanating arcs. Since the expected costs of downstream nodes would have already been calculated (the approach updates values starting from the destination and moving "backward"), this matrix would reduce to another matrix consisting of one row per unique cost. For instance, if arbitrarily $E[d] = E[e] = 1$ were assumed and taking $c_U = c^{c,d}{}_U = c^{c,e}{}_U$, $c_C = c^{c,d}{}_C = c^{c,e}{}_C$, with $c_U < c_C$, Relation (2.1) would reduce to

$$E^c = \begin{matrix} .75 \\ .25 \end{matrix}\begin{bmatrix} c_U + 1 \\ c_C + 1 \end{bmatrix}. \quad (2.2)$$

As this matrix can be evaluated as a single number (expected cost at node $b$), it can be used for further upstream calculations. This calculation is possible since the expected cost is not conditioned on additional arc states. Therefore, regardless of the previous path, the same expected value for successor nodes will be experienced. The matrix in relation (2.2) must be developed for the cost computations of node $b$ while calculating the minimum values over possible emanating arc states and the matrix can be discarded once this calculation is completed. However, if discarded, this matrix must be regenerated, if changes in the downstream expected values are made.

## 3. ALGORITHM DESIGN

Three algorithms are discussed: The first accounts for one-step arc dependencies, the second accounts for limited temporal dependence, and the third is a combination of the first two. Each algorithm is shown to perform polynomially with respect to the number of arcs, nodes, and states when applied to acyclic networks and probabilistic bounds are developed for general cases. The notation used is summarized in Table 1.

### *SD-OSP: One-step Spatial Dependence*

The SD-OSP algorithm is an adaptation of the all-to-one label-correcting algorithm [1] for deterministic, time-invariant shortest paths. There are two main differences:

1. In the label-correcting algorithm, each node has a single cost, which corresponds to the cost of the shortest path from the current node to the destination. In this recourse algorithm, each node $j$ has $\sum_{i\in\Gamma^{-1}(j)}|S(i, j)|$ costs, representing the expected cost depending on which predecessor arc was traversed and at what state, as illustrated in the example in the previous section.
2. The label-correcting algorithm computes the shortest path from all origins to a destination node. Each node is also labeled with a single successor node (*pathpointer)*. In the recourse algorithm, each node will have a *pathpointer* for each possible predecessor arc-state that might have been traversed. It chooses the *pathpointer* corresponding to the state traversed and follows this node.

This recourse algorithm is an all-to-one approach, meaning that the algorithm starts at the destination node and for each node computes the expected time to the destination given that the traveler traversed a predecessor arc to this node.

TABLE 1. OSP notation.

| | |
|---|---|
| $o$ | origin node |
| $t$ | terminal node |
| $E[b]$ | expected cost to the destination node from node $b$ |
| $E[b\|a, s]$ | expected cost to the destination node from node $b$, given that predecessor arc $(a, b)$ is traversed in state $s$ |
| $p_k^{a,b}$ | probability that arc $(a, b)$ is in state $k$ |
| $p_{s,k}^{a,b,c}$ | probability that arc $(b, c)$ is in state $k$, given that predecessor arc $(a, b)$ was in state $s$ |
| $S(a, b)$ | set of all possible states for the arc $(a, b)$ |
| $\|S(a, b)\|$ | number of possible states for the arc $(a, b)$ |
| $\|S\|$ | the maximum number of states over all arcs |
| $c_k^{a,b}$ | cost of arc $(a, b)$ in state $k$ |
| $SEL$ | scan eligible list |
| $\Gamma^{-1}(a)$ | set of all predecessor (upstream) nodes of $a$; $i \in \Gamma^{-1}(a) \Rightarrow$ arc$(i, a)$ exists |
| $\Gamma(a)$ | set of all successor (downstream) nodes of $a$; $j \in \Gamma(a) \Rightarrow$ arc$(a, j)$ exists |
| $\epsilon$ | vector of probabilities for possible states at a node (first column of matrix 2.1) |
| $\pi$ | vector of minimum costs for possible states at a node (second column of matrix 2.1) |

### *SD-OSP Algorithm Pseudo-Code*

For all $i\in\Gamma^{-1}(t)$, $s\in S(i, t)$, set $E[t|i, s]: = 0$
For all $j\in N/t$, $i\in\Gamma^{-1}(j)$, $s\in S(i, j)$, set $E[j|i, s]: = \infty$
$SEL: = \Gamma^{-1}(t)$
while $SEL\neq\varnothing$
  Remove an element $q$ from $SEL$
    for all $i\in\Gamma^{-1}(q)$ $s\in S(i, q)$, $j\in\Gamma(q)$

$$tempE[q|i, s]: = \sum_{k\in|S(q,j)} p_{s,k}^{i,q,j}(c_k^{q,j} + E[j|q, k])$$

    if $tempE[q|\, i, s] < E[q|i, s]$
      $E[q|i, s]: = tempE[q|i, s]$
      $pathpointer\,[q|i, s]: = j$
      $SEL: = SEL \cup \Gamma^{-1}(i)$
End

The online algorithm assumes that every node has a predecessor node. However, an origin node does not have a predecessor one; therefore, dummy nodes are added to the network. The arc from a dummy node to an origin node has one state with a total probability of 1. In the following simple example, the single dummy node and state are denoted $o'$ and $s'$, respectively.

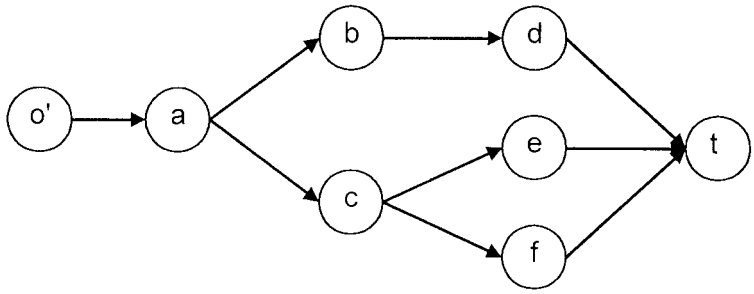


FIG. 3. Example network to demonstrate SD-OSP.

Consider the network in Figure 3 with probability matrices as indicated.

$$P^{o',a,b} = P^{o',a,c} = \begin{array}{c} \\ s' \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .5 & .5 \end{pmatrix} \end{array}$$

$$P^{a,b,d} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .5 & .5 \\ .5 & .5 \end{pmatrix} \end{array}$$

$$P^{a,c,e} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .8 & .2 \\ .2 & .8 \end{pmatrix} \end{array}$$

$$P^{a,c,f} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .3 & .7 \\ .7 & .3 \end{pmatrix} \end{array}$$

$$P^{b,d,t} = P^{c,e,t} = P^{c,f,t} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .6 & .4 \\ .4 & .6 \end{pmatrix} \end{array}$$

$a$ = origin node; $t$ = destination node.

$$c_1^{i,j} = 1 \quad \forall\, (i, j) \in A$$
$$c_2^{i,j} = 2 \quad \forall\, (i, j) \in A$$

The recourse algorithm performs seven iterations for this example. To illustrate the steps of the algorithm, Table 2 summarizes the intermediate results in every iteration. Each entry has the expected distance from the origin followed by the successor path pointer.

In iteration 1, the algorithm computes for $b \in \Gamma^{-1}(d)$, $\{1, 2\} \in S(b, d)$, $t \in \Gamma(d)$:

$$E[d|b, 1] = .6(1 + 0) + .4(2 + 0) = 1.4$$

$$E[d|b, 2] = .4(1 + 0) + .6(2 + 0) = 1.6$$

$$pathpointer\,[d|b, 1] = pathpointer\,[d|b, 2] = t.$$

In the last iteration, the expected distance from node $a$ to the destination node $t$ is computed using the dummy node and state:

From node b:

$$E[a] = E[a|o', s'] = .5(1 + 3) + .5(2 + 3) = 4.5$$

From node c:

$$E[a] = E[a|o', s'] = .5(1 + 2.64) + .5(2 + 3.24)$$
$$= 4.44.$$

Therefore, $pathpointer[a] = c$.

Since $E[a]$ from node $c$ is less than $E[a]$ from node $b$, a traveler's first decision when entering the network is to choose arc $(a,c)$. Once the traveler reaches node $c$, he or she will make a decision based on the state of arc $(a,c)$. If arc $(a,c)$ is in state 1, then the traveler will choose arc $(c,e)$ next; otherwise, arc $(c,f)$ will be selected.

In the previous example, the algorithm favors a solution consisting of a path set with the most nonoverlapping topological paths. This solution provides options to the traveler, thereby allowing further decisions to be made downstream depending on the state of $(a,c)$. In the example, there were two paths within the solution: *a–c–f–t* and *a–c–e–t*. The *a priori* solution of the above example can be calculated by enumerating all the paths. This *a priori* solution yields an expected cost of 4.5 since all three paths from node $a$ to $t$ have an expected cost of 4.5. However, the recourse solution has an expected cost of 4.44.

**Property 4.** *On an acyclic graph, the SD-OSP algorithm has computational complexity* $O(n^2\, m\, |S|^2)$.

**Corollary 1.** *At the* $k^{th}$ *pass through the complete node list, the algorithm will compute an optimal path set for all nodes that are connected to the source node by a shortest path set consisting of paths with k or fewer arcs given a first in first out (FIFO) SEL.*

**Proof.** The proofs of the above property and corollary follow from the general label-correcting proofs when a

TABLE 2. Spatial algorithm execution.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| *SEL* | $\{d, e, f\}$ | $\{e, f, b\}$ | $\{f, b, c\}$ | $\{b, c\}$ | $\{c, a\}$ | $\{a\}$ | $\varnothing$ |
| $E[a]$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 4.44 (*c*) |
| $E[b\|a, 1]$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 3 (*d*) | 3 (*d*) | 3 (*d*) |
| $E[b\|a, 2]$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 3 (*d*) | 3 (*d*) | 3 (*d*) |
| $E[c\|a, 1]$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 2.64 (*e*) | 2.64 (*e*) |
| $E[c\|a, 2]$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 3.24 (*f*) | 3.24 (*f*) |
| $E[d\|b, 1]$ | $\infty$ | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) |
| $E[d\|b, 2]$ | $\infty$ | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) |
| $E[e\|c, 1]$ | $\infty$ | $\infty$ | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) |
| $E[e\|c, 2]$ | $\infty$ | $\infty$ | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) |
| $E[f\|c, 1]$ | $\infty$ | $\infty$ | $\infty$ | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) | 1.4 (*t*) |
| $E[f\|c, 2]$ | $\infty$ | $\infty$ | $\infty$ | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) | 1.6 (*t*) |
| $E[t\|d, 1]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E[t\|d, 2]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E[t\|e, 1]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E[t\|e, 2]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E[t\|f, 1]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E[t\|f, 2]$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

FIFO SEL is used [1]. Unlike the label-correcting deterministic shortest path algorithm, each node maintains an expected cost representing the shortest path, not from a single path, but a path set. Furthermore, unlike the general label-correcting algorithm, each pass does not require $O(1)$ computations for each arc $(q,j)$. Instead, for each arc a possible expected cost must be computed for every predecessor node $i$ and state $s$:

$$\text{for all } i \in \Gamma^{-1}(q), s \in S(i, q), j \in \Gamma(q)$$

$$tempE[q|i, s] = \sum_{k \in S(q,j)} p_{s,k}^{i,q,j}(c_k^{q,j} + E[j|q, k]).$$

Each of these cost computations requires a summation over all states $S(q,j)$. If $|S|$ is taken to be the maximum number of states over all arcs and $n$ is the maximum degree of a node, this requires at most $n\,|S|^2$ computations for each arc, for each pass. This yields a final complexity of $O(K\,n\,m\,|S|^2)$, where $K$ is the number of passes; $K$ is equal to $n$ if the network contains no cycles which establishes the complexity of this algorithm. ∎

Since an optimal solution might include cycles, the number of passes required, $K$, is unknown. Furthermore, it has been shown that an optimal solution might include an arbitrarily large number of links in a path, which would suggest that $K$ might approach infinity. However, if all arc costs are assumed to be positive, performance bounds can easily be obtained, and if an error term is taken, a required number of passes can be calculated.

**Property 5.** *The maximum difference between the cost of the best solution after K passes and the optimal cost is bounded by* $(1 - F(K)) \times (n\Xi - \xi)$.

**Proof.** The solution consists of multiple state-paths with known costs and probabilities. As $K$ increases, certain state-paths are replaced with longer, in terms of number of arcs, state-paths with lower cost. However, after $K$ passes, it is known that all remaining unaccounted paths have a summed probability of $1 - F(K)$. Furthermore, the maximum cost reduction that can be experienced from a suboptimal path being replaced by an optimal one is $n\Xi - \xi$. ∎

Let $\alpha$ be an acceptable percentage error term; since $\xi$ is known to be a lower bound on the cost of an optimal solution, a relationship for the required number of passes can be expressed as

$$(1 - F(K)) \times (n\Xi) - \xi)/\xi \leq \alpha.$$

From Property 3, it is known that after $K$ passes the probability that a path longer than $K$ is unaccounted for is bounded above by $n\Xi|\ (\xi K)$. This leads to the following corollary:

**Corollary 2.** *For an acceptable maximum error term* $\alpha$, *the required number of passes* $K$ *is given by* $K \geq (n\Xi(n\Xi - \xi))/(\xi^2\alpha)$.

For given costs and $\alpha$, this yields a final complexity of $O(n^2 m\, S^2\, \Xi\ (n\ \Xi\ -\ \xi.)|\ (\xi.2\ \alpha))$.

### *TD–OSP: Temporal Dependence Algorithm*

This section presents an algorithm for the shortest path with recourse accounting for limited temporal dependence. Similar to the spatial algorithm discussed above, this is an all-to-one label-correcting algorithm. However, unlike the spatial recourse algorithm, each node has a single cost

value, which describes the cost of the shortest path from that node to the destination and no path pointer information is stored.

*Online Path Construction*

For the temporal recourse algorithm, it is quite difficult to find an *a priori* path solution. Instead of storing path pointer information required for this *a priori* solution within the temporal recourse algorithm, the approach presented here constructs a path online, as a traveler moves through a network. A solution to the TD-OSP will consist of a set of expected costs for each node. When a traveler approaches a node, the arc costs for all emanating arcs are learned. At this point, the computation of the next arc is equivalent to the deterministic shortest path for a single-node cost computation, since the downstream node expected cost is known and is not dependent on this information. Therefore, the newly learned arc costs can be added to each expected value, and the resulting minimum cost can be chosen as the subsequent arc.

*Computation of the Expected Cost Matrix*

An important aspect of this algorithm is the calculation of the expected cost matrix (2.1). If done in a straightforward manner, this would mean that the probability of every state for every arc is multiplied by every other arc-state probability, resulting in an exponential number of calculations ($|S|^{|A|}$). However, this same calculation can be performed in a more efficient way since information regarding the path pointer need not be maintained.

For any pairwise combination of the probability matrices relating to arcs $(a,b)$ and $(a,c)$, the resulting matrix can have at most a number of different states equal to the sum of the number of different states of arcs $(a,b)$ and $(a,c)$. This is because the only computation performed is minimization; therefore, the number of different output values cannot exceed the number of different input numbers. This will result in a matrix which has at most $|S(a,b) + S(a,c)|$ rows, instead of $|S(a,b)|$ times $|S(a,c)|$rows. For example, take a node $(a)$ which has three downstream nodes $(b,c,d)$ and probability matrices in the following form:

$$P^{a,b} = \begin{matrix}.333\\.333\\.333\end{matrix}\begin{bmatrix}1\\4\\8\end{bmatrix},\ P^{a,c} = \begin{matrix}.5\\.5\end{matrix}\begin{bmatrix}2\\6\end{bmatrix},\ P^{a,d} = \begin{matrix}.5\\.5\end{matrix}\begin{bmatrix}3\\5\end{bmatrix}.$$

If the expected cost matrix were calculated in a straightforward fashion, it would have 12 rows and one column. However, if only the first two probability matrices are combined, a table is initially formed of size 6 by 1:

$$\begin{matrix}.166\\.166\\.166\\.166\\.166\\.166\end{matrix}\begin{bmatrix}\min(1+E[b],\,2+E[c])\\ \min(1+E[b],\,6+E[c])\\ \min(4+E[b],\,2+E[c])\\ \min(4+E[b],\,6+E[c])\\ \min(8+E[b],\,2+E[c])\\ \min(8+E[b],\,6+E[c])\end{bmatrix}.$$

Regardless of the values chosen for $E[b]$ and $E[c]$, there can exist at most five different values within this matrix. Since path pointer information is not stored, and only the expected cost is needed, this matrix can be reduced to one with five rows without the loss of information. This 5 by 1 matrix can then be combined with the one from $p^{a,d}$ to form an initial matrix of size 10 by 1, which can subsequently be reduced to a 7 by 1 matrix. If simple matrix copying with the elimination of redundant information through a binary row search is used to perform this reduction, the computational effort for each matrix reduction is bounded by the logarithm of the maximum number of rows for the matrix.

***TD-OSP Algorithm Pseudo-Code***

Set $E[t] := 0$
for all $i \in N/t$, let $E[i] := \infty$
$SEL := \Gamma^{-1}(t)$
while $SEL \neq \varnothing$
  Remove an element $q$ from $SEL$
  $\varepsilon := [1]$
  $\pi := [\infty]$
  for all $j \in \Gamma(q)$
    $temp\varepsilon := [\varnothing]$
    $temp\pi := [\varnothing]$
    $r := 0$
    for all $k \in S(q,j)$
      for all $l = 1, \ldots, |\varepsilon|$
        $temp\varepsilon_r := \varepsilon_l P_k^{q,j}$
        *if* $c_k^{qj} + E[j] < \pi_l$
          $temp\pi_r := c_k^{qj} + E[j]$
        else
          $temp\pi_r := \pi_l$
        $r := r + 1$
    $[\varepsilon, \pi] :=$ Reduce($temp\varepsilon$,$temp\pi$)
  $tempE[q] := \sum_{\forall k \in |\varepsilon|} \varepsilon_k \pi_k$
  if $tempE[q] < E[q]$
    $E[q] := tempE[q]$
    $SEL := SEL \cup \Gamma^{-1}(q)$
End
  Function Reduce($temp\varepsilon$, $temp\pi$)
    $\varepsilon := [\varnothing]$
    $\pi := [\varnothing]$
    for $k = 1, \ldots, |temp\pi|$
      $j :=$ Search($\pi$,$temp\pi_k$)
      $\pi_j := temp\pi_k$
      $\varepsilon_j := temp\varepsilon_k + \varepsilon_j$
    return($\varepsilon$, $\pi$)
End

Here, *Search()* performs a search on vector $\pi$ for value $temp\pi_k$ and returns the location if found. If not found, it returns the first unused location within vector $\pi$.

Next, the following property of TD-OSP solution is stated and proved:

**Property 6.** *For an acyclic graph, the TD-OSP has computational complexity* $O(n^2 m\ |S|^2 \log(|S|\ n))$.

**Proof.** This proof follows that for the previous spatial case except for the computational complexity of a single pass. If $|S|$ is taken to be the maximum number of states over any arc, then any arbitrary combined matrix cannot have size exceeding $|S|\ n$ since this is the maximum number of different state costs from $n$ matrices. If this largest possible matrix were combined with another arc cost matrix of maximum size $|S|$, at most $|S|^2\ n$ computations will be performed. Therefore, no more than $|S|^2\ n$ computations will be performed for any single arc when combining probability matrices.

After the matrices are combined, the resulting expected cost matrix must be reduced. One way of performing this *Reduce()* function for a matrix of maximum size $|S|^2\ n$ would be to use a Binary Search implementation as follows:

i. Create an empty Dynamic Structure (DS).
ii. Remove row ($temp\pi$, $temp\varepsilon$), consisting of a state *cost* and *probability*, from the original matrix.
iii. Perform a Binary Search on DS for the cost of ($temp\pi$).
iv. If it exists, add the probability ($temp\varepsilon$) to the same element in the DS.
v. If it does not exist, insert ($temp\pi$, $temp\varepsilon$) into the DS at the place pointed to by the binary search.

This procedure must be repeated until the original matrix is empty, whose size is bounded by $|S|^2\ n$ since $n$ is the maximum number of arcs emanating from a node. The most computationally expensive part in this procedure is step iii. Since the maximum size of the DS is equal to the number of different states in the original matrix, it cannot grow larger than $|S| + |S|\ n$. Therefore. the binary search would require at most $O(\log(|S|\ n))$ computations. Since this overall procedure must be repeated $|S|^2\ n$ times, this gives a final complexity of $O(|S|^2\ n \log(|S|\ n))$ for the matrix reduction associated with each arc.

Therefore, the algorithm complexity for each arc computation is $O(|S|^2\ n + |S|^2\ n \log(|S|\ n))$ or $O(|S|^2\ n \log(|S|\ n))$. Since these computations must be performed for each arc and for each pass, the final complexity will be $O(K\ n\ m\ |S|^2 \log(|S|\ n))$, where $K$ is the number of passes; $K$ is bounded by $n$ if the network does not contain cycles. Corollary 2 applies as a bound on the number of passes in this model as well, and cost bounds will therefore follow the ones stated for the spatial case. ∎

*TSD–OSP: Temporal-Spatial Dependence Algorithm*

This section introduces an algorithm for the shortest path with recourse while accounting for limited temporal and spatial dependency. This algorithm combines elements of both of the previous approaches. In this problem variant, the costs of all emanating arcs $(a,b)$ are learned once node $a$ is reached. Furthermore, this yields improved spatial information for all arcs emanating from each node $b$. This problem variation might be suitable for certain routing applications, since it describes the increasing uncertainty related to proximity: Once a node is reached, the cost of the immediate arcs are learned with virtual certainty, while those one step away are learned with improved, but imperfect, confidence.

As in the SD-OSP, multiple costs are maintained for each predecessor node and state, and as in the TD-OSP, no path information is maintained and the expected cost matrix must be developed over all possible successor arc-states. Since no path information is stored, the route must be constructed online as in the temporal case. Here, the conditional expected cost ($E[n|S|i,j]$ ) is used for this construction. The *Reduce()* subroutine is also employed as defined in the temporal problem variant.

***TSD-OSP Algorithm Pseudo-Code***

for all $i \in \Gamma^{-1}(t)$, $s \in S(i, t)$, set $E[t|i, s] := 0$
for all $q \in N/t$, $i \in \Gamma^{-1}(q)$, $s \in S(i, q)$, set $E[q|i, s] := \infty$
$SEL := \Gamma^{-1}(t)$
while $SEL \neq \varnothing$
  Remove an element $q$ from $SEL$
  for all $i \in \Gamma^{-1}(q)$
    for all $s \in S(i, q)$
      $\varepsilon := [1]$
      $\pi := [\infty]$
      for all $j \in \Gamma(q)$
        $temp\varepsilon := [\varnothing]$
        $temp\pi := [\varnothing]$
        $r := 0$
        for all $k \in S(q, j)$
          for all $l = 1, \ldots, |\varepsilon|$
            $temp\varepsilon_r := \varepsilon_l P^{i,q,j}_{s,k}$
            if $c^{qj}_k + E[j|q, k] < \pi_l$
              $temp\pi_r := c^{qj}_k + E[j|q, k]$
            else
              $temp\pi_r := \pi_l$
            $r := r + 1$
        $[\varepsilon, \pi] :=$ Reduce($temp\varepsilon$, $temp\pi$)
      $tempE[q] := \sum_{k \in |\varepsilon|} \varepsilon_k \pi_k$
      if $tempE[q] < E[q|i, s]$
        $E[q|i, s] := tempE[q]$
        $SEL := SEL \cup \{i\}$
End

**Property 7.** *On an acyclic graph, the TSD-OSP algorithm has computational complexity* $O(n^3 m|S|^3 \log(|S|\ n))$.

This follows from the previous complexity results for acyclic graphs and Corollary 2 that establishes a probabilistic bound on the number of passes, as well as the cost bounds developed for the spatial and temporal cases.

While this section discussed the combined limited dependency model, it also demonstrated how this approach

can be applied accounting for additional dependency steps. Spatial dependency can be extended for multiple arcs, but would require an exponential increase in terms of computation and storage with regard to the number of steps. Furthermore, accounting for additional steps of dependency should only improve the expected performance of the online approach.

## 4. COMPUTATIONAL TESTS

The computational results in this section aim to support the previously stated theoretical findings as well as to evaluate numerically the performance of the TD-OSP algorithm. We demonstrate that the stated complexity for the TD-OSP is a loose worst-case bound. In addition, the precision of commercially available computers, typically limited to 32 or 64 bit, forces cycling to terminate in a relatively small number of iterations, reducing even further the computational time requirements. In other words, while continuing to cycle theoretically yields some small benefit, after a fairly limited number of iterations, the difference is beyond the precision of the machine and the cycling terminates. This does not appear to be problematic, since most numerical approaches typically ignore errors in excess of what can be captured by a 64-bit machine, and for most practical shortest path implementations, additional precision would result in a negligible benefit. Furthermore, if such cycling does incur excessive computational time, the stopping criterion developed in Corollary 2 can be employed for a specified acceptable error.

Another important insight gained from the computational results is the quantification of the benefits of an online approach versus an off-line (purely *a priori*) one; this is possible only computationally, since it is difficult to derive closed-form expressions for their comparative performance. To shed some light on the potential advantages of online approaches, a series of random experiments are conducted in this section. The temporal algorithm is implemented rather than the spatial or combined, due to its lower data requirements. Since no spatial dependencies need be generated, the tests should be somewhat less sensitive to the means by which the data is produced. For each experiment, the temporal online algorithm is compared with a traditional deterministic shortest path algorithm (equivalent to the *a priori* solution for this problem variant). The primary question, however, is not that of computational time but the expected costs of the solutions; the deterministic version is clearly much faster than is the online algorithm. However, it should be noted that the average running time of the online algorithm was less than a second and reached a maximum of approximately 80 seconds for certain large problem instances. These tests were conducted on a rather low end Pentium III 500-MHz machine with 256 megabytes of memory running the Linux operating system.

The algorithm is implemented in C++ and tested on 200 randomly generated networks for each experiment. The network attributes such as connectivity, size, and cost variance are specified in the discussion of each experiment. The networks are created with each arc having a discrete uniform probability distribution for the cost with up to 10 such possible costs. Each experiment compares the difference between the average expected costs of all shortest paths, as determined by the deterministic and online shortest path algorithms, to a randomly chosen destination. It should be noted that since no spatial dependency will be modeled in these experiments, the deterministic case is equivalent to the *a priori* stochastic shortest path case. In other words, the *a priori* stochastic shortest path without spatial dependency can be found by replacing each arc cost with the expected value. This is an additional justification for examining the temporal version of the algorithm at this time, since the true benefit of accounting for recourse can be examined separately from the benefit realized from accounting simply for stochasticity. In the spatial case, the *a priori* stochastic shortest path is not identical to the deterministic one. Furthermore, the means of evaluating the spatial algorithm is not perfectly clear since the dependencies must be developed. These interarc dependencies may differ by application and for a given application, such as traffic routing, might vary greatly by road type. Therefore, the computational analysis will focus on the temporal case, since the required input data and numerical results appear to be more general.

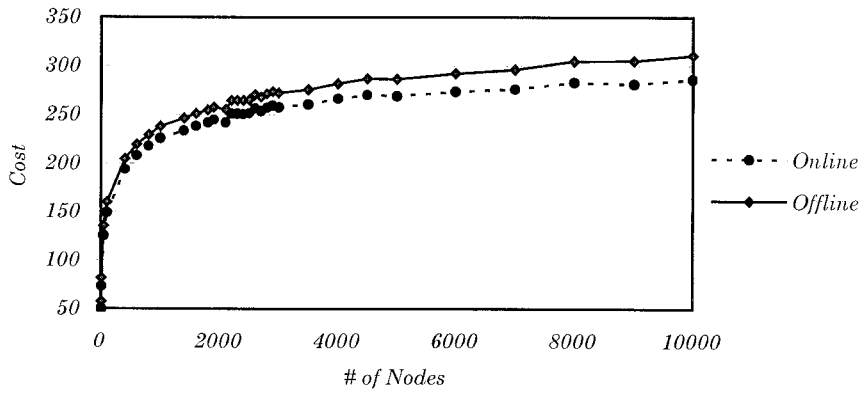


FIG. 4. Impact of problem size on average cost.

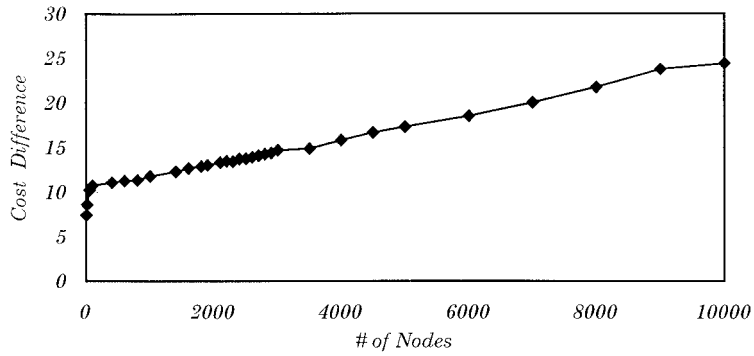


FIG. 5. Impact of size on cost difference.

*Impact of Network Size*

The first set of experiments examines the impact of an increased network size on the expected cost differences. For this set, the average node degree is four, and the arc costs range from one to 100. Figure 4 shows the average cost of the two algorithms between all origins and the same randomly chosen destination over the 200 randomly selected samples.

Figures 5 and 6 depict how the absolute cost difference and percentage difference between the two algorithms change with the size of the network. In all cases, the online approach generates a lower expected cost. As shown in Figure 5, the absolute cost difference increases in favor of the online approach with the network size. This result seems intuitive, since at any decision point, the expected cost related to the online solution will be lower than that corresponding to the off-line algorithm. Therefore, as the average path size and number of decision points increase, where each decision point has a lower online than off-line expected cost, the difference in costs should tend to increase. However, the percentage difference represents a potentially more important measure for many applications, and while it is clear that the cost difference will increase, the rate of change is not strictly increasing with the size of the problem. Figure 6 illustrates that the percentage difference appears to increase with network size in the longer term.

For smaller networks, however, the percentage benefit decreases rapidly. This may be due to the dominant role that stochasticity is playing in small networks. If the average path length is only a few links, then the choice of the next link represents a major portion of the overall path length. As the average path length and cost increase, each decision point has less of an impact on the overall cost. This would explain the initial decrease in percentage benefit. As the network size continues to grow, however, the average costs themselves (Fig. 4) begin to grow less rapidly. Therefore, the absolute cost difference behavior, shown in Figure 5, might begin to dominate and result in the percentage difference increase.

*Impact of Maximum Cost and Cost Variance*

In this experiment, the network size is 1000 nodes with an average degree of four. First, a series of experiments are performed with the maximum costs varying from 10 to 2000 and the minimum of the costs being 1. As shown in Figure 7, relatively little difference was observed for this set of experiments.

An additional experiment is run where the average arc cost remains 50, but the minimum/maximum values vary. As shown in Figure 8, the cost variance has a very noticeable impact on the benefit from the online algorithm. As the variance of the cost increases, the percentage benefit also increases. This is quite intuitive, since the online algorithm tends to perform well when the costs are uncertain. One extreme case would be the zero variance case (or determin-

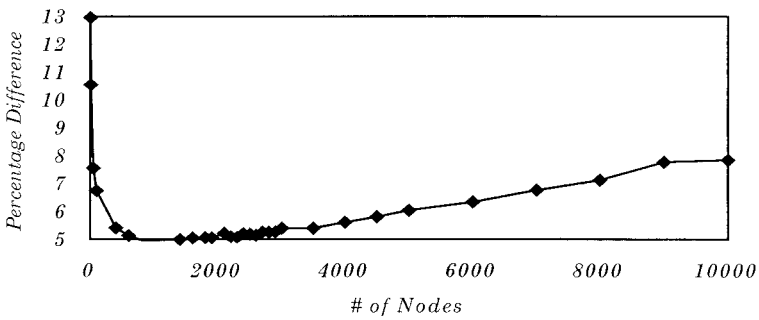


FIG. 6. Impact of size on percentage benefit.

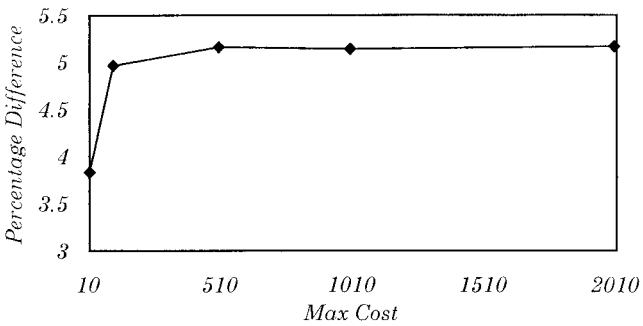


FIG. 7. Impact of maximum cost on percentage benefit.

istic costs), where the online and off-line (*a priori)* algorithms perform identically.

*Impact of Connectivity*

The final experiment examines the impact of the average degree on the percentage benefit related to the online algorithm. For this experiment, the network size is 1000 nodes and the arc costs range from one to 100. As shown in Figure 9, the percentage benefit increases in favor of the *online* algorithm with the average degree of a node. This is intuitive, since as the potential decision options increase, the online algorithm can more readily exploit the additional opportunities than the off-line algorithm.

Therefore, the online costs are reduced as the number of alternatives increase. In this specific case, a substantial benefit of nearly 45% is realized with an average degree of 50.

## 5. CONCLUSIONS

This paper was concerned with the time-invariant online SPP that accounts for limited forms of spatial and temporal dependency. General properties and specific cases were examined to obtain insights into the behavior of the online SPP, such as cycling and the relationship of the *a priori* versus online solution. The impact of information, here modeled in the form of local spatial and temporal dependencies, was shown to yield differences in the solution and improvements in the expected cost of the online solution.

Numerical results on limited test networks verified some

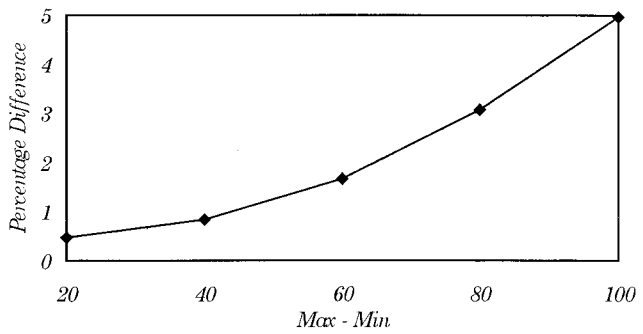


FIG. 8. Impact of cost variance on percentage benefit.

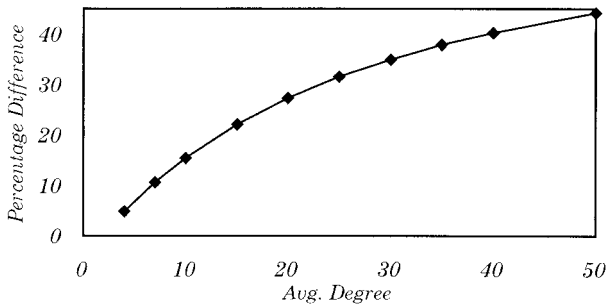


FIG. 9. Impact of connectivity on percentage benefit.

of the theoretical findings and demonstrated that online optimum paths outperform offline shortest paths by up to 40% under certain conditions. While the benefits depend on the network topology and cost structure, the cost difference exceeded 5% for typical routing problems in nearly all examples tested for this paper. Furthermore, it was found that benefits tend to increase with network size, node degree, and cost variance.

Polynomial recourse algorithms for acyclic problems were developed that lead to a solution with a cost equal to or better than the *a priori* versions. Complexity bounds were developed for cyclic problems where all arc costs are greater than zero and an optimal cost error term is used. Clearly, while this analysis provides insight and potential tools for the online SPP, it could be further extended to account for networks with additional forms of information and time dependency.

## Acknowledgments

The authors are grateful to the editor-in-chief, the associate editor, and an anonymous referee for their very constructive comments. The contents of the paper remain, of course, the sole responsibility of the authors.

## REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms, and applications, Prentice-Hall, Upper Saddle River, NJ, 1993.

[2] G. Andreatta and L. Romeo, Stochastic shortest paths with recourse, Networks 18 (1988), 193–204.

[3] J.S. Croucher, A note on the stochastic shortest route problem, Nav Res Log Q 25 (1978), 729–732.

[4] L. Fu and L.R. Rilett, Expected shortest paths in dynamic and stochastic traffic networks, Trans Res Part B Method 32 (1998), 499–516.

[5] R. Hall, The fastest path through a network with random time-dependent travel times, Transp Sci 20 (1986), 182–188.

[6] E. Miller-Hooks and H.S. Mahmassani, Least expected time paths in stochastic, time-varying transportation networks, Trans Sci 34 (2000), 198–215.

[7] G.H. Polychronopoulos and J.T. Tsitsiklis, Stochastic shortest path problems with recourse, Networks 27 (1996), 133–143.