Using Stochastic-Dominance Relationships for Bounding Travel Times in Stochastic Networks*

Chao-Lin Liu
Department of Computer Science
National Chengchi University
Mucha, Taipei 116, Taiwan
chaolin@cs.nccu.edu.tw

Michael P. Wellman
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
wellman@umich.edu

Abstract

We consider stochastic networks in which link travel times are dependent, discrete random variables. We present methods for computing bounds on path travel times using stochastic dominance relationships among link travel times, and discuss techniques for controlling tightness of the bounds. We apply these methods to shortest-path problems, show that the proposed algorithm can provide bounds on the recommended path, and elaborate on extensions of the algorithm for demonstrating the anytime property.

1 Introduction

A main goal of intelligent transportation systems is to improve efficiency in transportation systems by providing useful information to travelers. The travel time on a particular path itself can be of interest to travelers for trip planning. A more interesting application of travel times is in route-guidance systems that compare travel times on alternative paths to find the path that meets the demand of travelers.

The literature has seen a wide variety of algorithms applicable to finding desired paths for travelers [1, 5, 7]. Many of these algorithms assume constant cost, such as geometric distance, for traveling between connected locations. In contrast, link travel times are rarely constant, and should be treated as random variables in actual transportation networks. Moreover, in these so-called "stochastic networks," the probability distributions over link travel times also depend on the time travelers enter the links. Such dependent relationships among link travel times make standard shortest-path algorithms such as Dijkstra's algorithm [6] inapplicable to shortest-path problems in stochastic networks[9].

In recent years, researchers have found search algorithms for shortest-path problems in stochastic networks. Kaufman and Smith study networks in which link travel times are constants that depend on the time when travelers enter the links. They show that Dijkstra's algorithm remains applicable for shortest-path problems when link travel times in the network have the *consistency* property [10]. Wellman et al. generalize the consistency property, and consider networks in which link travel times are dependent random variables. They show that standard search algorithms can find the shortest path when link travel times have the *stochastic consistency* property [12].

In this paper, we study the problem of computing travel times in stochastic transportation networks in which link travel times are nonnegative and integer-valued random variables. We consider situations in which computation time is insufficient for obtaining exact probability distributions over travel times of interest. To this end, we propose methods that take advantage of stochastic consistency for computing bounds on probability distributions over travel times, and we show that these bounds are guaranteed to tighten if more time is allocated for computation. We then propose an algorithm that applies these approximation methods to finding the shortest path in stochastic networks. We show that our algorithm can compute the maximum deviation of the distribution over the travel time on the path selected by our algorithm from the distribution over the travel time on the actual shortest path.

In the following section, we review the concept of stochastic dominance, and define bounds on distributions over travel times. In Section 3, we discuss methods that employ the stochastic-dominance relationships for computing lower and upper bounds on travel times. We then present and analyze our algorithm that employs these bounds to shortest-path problems, and extend the algorithm for the anytime property in Section 4. We conclude with a brief discussion of further extensions of the algorithm.

2 Notation and definitions

Let O and D denote the origin and destination of the traveler, respectively. We denote a path *P* that allows the traveler to go from O to a location Y via intermediate locations $L_1, L_2, \ldots$, by $O \to L_1 \to L_2 \to \ldots \to Y$. We denote the arrival time at location Y via path *P* by the random variable T_Y, leaving "P" implicit. When necessary, we use a superscript i to distinguish travel costs of different paths P_i, e.g., T_Y^1 and T_Y^2. We call a possible value t_Y of T_Y a *state* of T_Y, and denote the set of all possible states of T_Y by $V(T_Y)$. When necessary,

*Appeared in the *Proceedings of the IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems.* Tokyo, Japan, 1999.

we use an extra subscript of t_Y to distinguish different states of T_Y, e.g., $t_{Y,1}$ and $t_{Y,2}$. We assume that the traveler will not stay at intermediate locations, so we do not differentiate arrival times and departure times for intermediate locations.

Let $F(x)$ and $F'(x)$ be two cumulative distribution functions (CDF) of a random variable X. We say that $F(x)$ *first-order stochastically dominates* $F'(x)$ if

$$F(x) \leq F'(x), \text{ for all states } x \text{ of } X, \tag{1}$$

and we denote this relationship by $F(x)$ FSD $F'(x)$ [13].

Given $F(t_Y^1)$ FSD $F(t_Y^2)$, we can infer that the probability of arriving at Y by time z via path P_2 must be larger than that via path P_1. As a result, the expected arrival time at Y on path P_1 is later than that on path P_2. In fact, it can be shown [8] that (1) holds if and only if for all monotonically increasing functions $g(x)$,

$$\int g(x)dF(x) \geq \int g(x)dF'(x). \tag{2}$$

We define bounds on probability distributions over travel times based on stochastic dominance relationships. Consider a path to a location Y and the arrival time T_Y. We say that a CDF, denoted $F^U(t_Y)$, is an *upper bound* of $F(t_Y)$ if $F^U(t_Y)$ FSD $F(t_Y)$. An upper bound on the distribution over T_Y represents a worst-case scenario of the travel time on the path. The actual probability of arriving at Y by *any* t_Y, i.e., $F(t_Y)$, is not smaller than the upper bound $F^U(t_Y)$, as illustrated in the following figure. Analogously, we say that a CDF, denoted $F^L(t_Y)$, is a *lower bound* of $F(t_Y)$ if $F(t_Y)$ FSD $F^L(t_Y)$.

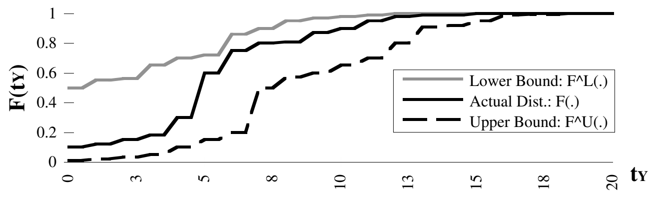

Figure 1: Bounds on distributions over travel times.

From the viewpoint of stochastic dominance, we define the relative goodness among paths. As we have explained, $F(t_Y^j)$ FSD $F(t_Y^i)$ implies that the probability of arriving at Y by any t_Y on P_i is not smaller than that on P_j. Assuming that travelers always prefer to arrive at their destinations earlier, we say that path P_i is *better* than path P_j. Applying this definition, P_k is better than P_l if an upper bound $F^U(t_X^k)$ of $F(t_X^k)$ dominates a lower bound $F^L(t_X^l)$ of $F(t_X^l)$. Analogously, we say that a path is *worse* than the other when the stochastic dominance relationships reverse.

3 State-space aggregation

Consider the problem of specifying distributions over arrival times T for the currently last location on a path. When the probability distributions over link travel times are specified by discrete distributions, the number of states of T increases with the number of links in P. Assume that, given a departure time from one terminal of a link, there are k possible arrival time at the other terminal, and that we add at most α states to T when we append a link to P. The following figure shows the growth of the number of states of T when the departure time from O is known.

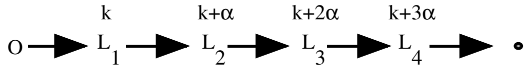

Figure 2: The number of states of T increases with the number of links in P.

Consequently, the time required to compute travel times increases rapidly with the number of links in a path. In this example, computing the distribution of T_{L_4} from that of T_{L_3} with the following formula[1]

$$\Pr(t_{L_4}) = \sum_{t \in V(T_{L_3})} \Pr(t_{L_4} | T_{L_3} = t) \Pr(T_{L_3} = t)$$

will require $(k+2\alpha)(k+3\alpha)$ multiplications. We need to carry out $(k+2\alpha)$ multiplications for each of the $(k+3\alpha)$ states of T_{L_4} in the worst case. Extending this reasoning for a path with N links, the computational complexity of computing the distribution of T is $O(\alpha^2 N^3)$.

One way to cope with the increasing computational costs that result from extending P is to aggregate consecutive states of T. For instance, we may aggregate the first three states $t_{L_i,0}, t_{L_i,1}$, and $t_{L_i,2}$ of T_{L_i} into a new state $t_{L_i,[0,2]}$, and replace these original states with the aggregated state. Assume that we limit the number of states of T to β before we add a new link to P, and that there can be at most $(\gamma+\beta)$ states in T immediately after we append a link to P (but before we replace these $(\gamma+\beta)$ states by β aggregated states). We can derive that the computational complexity for this approximation method will be $O(\beta(\beta+\gamma)N)$ for a path with N links, provided that the cost for aggregating states of T is negligible.

Since γ must be no bigger than α, we can manipulate the value of β so that computing travel times with approximated distributions is computationally cheaper. In the following subsections, we present the idea in formal terms. We replace T_{L_i} with $\hat{T}_{L_i}$ after we aggregate the states of T_{L_i}. We denote approximate or assigned probability functions with an extra "$\hat{}$", e.g., "$\hat{\Pr}(\cdot)$" and "$\hat{F}(\cdot)$", and we denote the set of original states included in an aggregated state $\hat{t}$ by the membership function $m(\hat{t})$.

3.1 Aggregating states

We aggregate consecutive states of T to reduce the computational costs for computing distributions over travel times. Assume that L_i is currently the last location on the path. The aggregation of states of T_{L_i} requires two types of operations. Assume that we have obtained an approximate distribution of T_{L_i}, $\hat{\Pr}(t_{L_i})$, and that we have not aggregated the states of T_{L_i}. We need to assign the probability value of $\hat{\Pr}(\hat{t}_{L_i})$ for each state $\hat{t}_{L_i}$ of $\hat{T}_{L_i}$, where $\hat{t}_{L_i}$ is a state of $\hat{T}_{L_i}$ and $\hat{t}_{L_i}$ consists of a set of original, consecutive states of T_{L_i}. We also

[1] Any path cost must depend on the departure time o from O. For simplicity, we do not show this dependency henceforth.

need to assign the probability value of $\hat{\Pr}(t_{L_{i+1}}|\hat{t}_{L_i})$ when we add the link $L_i \rightarrow L_{i+1}$ into *P*.

Applying Bayes' theorem, we can find and set $\hat{\Pr}(\hat{t}_{L_i})$ and $\hat{\Pr}(t_{L_{i+1}}|\hat{t}_{L_i})$ such that we preserve the distributions for T_{L_j} of all locations L_j on the path [4]. These methods, however, require nontrivial calculations and can offset the reduction of computational costs. Consequently, we cannot apply these methods.

There is an intuitive and simple assignment for $\hat{\Pr}(\hat{t}_{L_i})$. Namely, we set the probability of arriving at L_i during a duration U to the sum of the probabilities of arriving at L_i during subintervals of U :

$$\hat{\Pr}(\hat{t}_{L_i}) = \sum_{t_{L_i} \in m(\hat{t}_{L_i})} \hat{\Pr}(t_{L_i}). \tag{3}$$

Given that we choose to use (3), we still have freedom in assigning values of $\hat{\Pr}(t_{L_{i+1}}|\hat{t}_{L_i})$. In fact, we control the directions of the bounds by choosing different ways to assign these probability values.

When link travel times have the stochastic consistency property as is defined by Wellman et al. [12], we obtain a lower bound on the travel time if we assign $\hat{\Pr}(t_{L_{i+1}}|\hat{t}_{L_i})$ such that

$$\hat{F}(t_{L_{i+1}}|\hat{t}_{L_i}) = \max_{t_{L_i} \in m(\hat{t}_{L_i})} F(t_{L_{i+1}}|t_{L_i}), \tag{4}$$

where $F(t_{L_{i+1}}|t_{L_i})$ represents the cumulative distribution over the traversal time on link $L_i \rightarrow L_{i+1}$, and can be found in the specification of the given transportation network. We obtain an upper bound on the travel time if we assign $\hat{\Pr}(t_{L_{i+1}}|\hat{t}_{L_i})$ such that

$$\hat{F}(t_{L_{i+1}}|\hat{t}_{L_i}) = \min_{t_{L_i} \in m(\hat{t}_{L_i})} F(t_{L_{i+1}}|t_{L_i}). \tag{5}$$

Both (4) and (5) can be simplified if the transportation network is stochastically consistent. A transportation network is stochastically consistent if and only if the probability of arriving at a location L_k by any given time cannot be increased by leaving current location L_j later [12]. Namely, let $u < v$ be departure times from L_j, we have

$$\Pr(T_k \leq t_k|T_j = u) \geq \Pr(T_k \leq t_k|T_j = v).$$

Therefore, a network is stochastically consistent if and only if $F(t_{L_{j+1}}|t_{L_j})$ of all links are non-increasing functions of t_{L_j}, and (4) can be simplified to the following assignment method:

$$\hat{F}(t_{L_{i+1}}|\hat{t}_{L_i}) = F(t_{L_{i+1}}|T_{L_i} = r), \tag{6}$$

where r is the earliest original state in $m(\hat{t}_{L_i})$. Analogously, (5) can be replaced by (6) by setting r to the latest original state in $m(\hat{t}_{L_i})$.

3.2 Illustration of state-space aggregation

Consider the three-location path shown in the following figure. Notice that distributions over link travel times are stochastically consistent. Assuming that the departure time from O is 0, we can read the distribution over the arrival time

$O \longrightarrow A \longrightarrow B$

Pr(TA=1|O=0)=Pr(TB=2|TA=1)=0.3 Pr(TB=3|TA=2)=0.4 Pr(TB=4|TA=3)=0.5
Pr(TA=2|O=0)=Pr(TB=3|TA=1)=0.4 Pr(TB=4|TA=2)=0.2 Pr(TB=5|TA=3)=0.3
Pr(TA=3|O=0)=Pr(TB=4|TA=1)=0.3 Pr(TB=5|TA=2)=0.4 Pr(TB=6|TA=3)=0.2

Figure 3: Illustration of state-space aggregation

$\hat{\Pr}(T_A)$ at location A from the figure. Since there is no approximation operation for obtaining arrival time at A, we have $\hat{\Pr}(T_A) = \Pr(T_A)$.

Assume that we apply (3) and (5) to aggregate the states of A into two new states: [1,2] and 3. Applying (3), we have $\hat{\Pr}(\hat{T}_A = [1,2]) = 0.3 + 0.4 = 0.7$ and $\hat{\Pr}(\hat{T}_A = 3) = 0.3$. Applying (5), we compute the new CDF for T_B given the new state space of T_A, and obtain the following probability values: $\hat{\Pr}(T_B = 3|\hat{T}_A = [1,2]) = 0.4$, $\hat{\Pr}(T_B = 4|\hat{T}_A = [1,2]) = 0.2$, and $\hat{\Pr}(T_B = 5|\hat{T}_A = [1,2]) = 0.4$.

Putting these numbers together, we can compute $\hat{\Pr}(t_B)$ for all t_B. For instance, $\hat{\Pr}(t_B = 4)$ is equal to the sum of $\hat{\Pr}(t_B = 4|\hat{T}_A = [1,2])\hat{\Pr}(\hat{T}_A = [1,2])$ and $\hat{\Pr}(t_B = 4|\hat{T}_A = 3)\hat{\Pr}(\hat{T}_A = 3)$. Notice that T_B has five states, whereas T_A has only three states.

We plot $\hat{F}(t_B)$ and $F(t_B)$ against values of T_B in the following chart. We can compare the curves and verify that $\hat{F}(t_B)$ is an upper bound of $F(t_B)$.

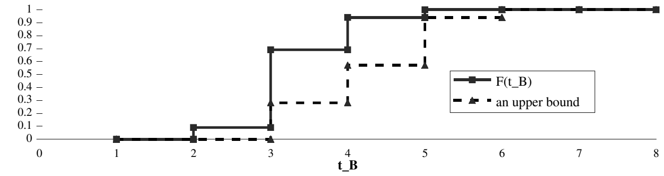

Figure 4: $F(t_B)$ and its upper bound

3.3 Proof

We show, by induction, that we obtain lower bounds on travel times if we employ (3) and (4) in state-space aggregation. We can show the case for obtaining upper bounds by employing (3) and (5) analogously.

First we examine $\hat{\Pr}(t_{L_1})$. For a given departure time from the origin O, $\hat{\Pr}(t_{L_1}) = \Pr(t_{L_1})$, so $\hat{F}(t_{L_1})$ represents an lower bound of $F(t_{L_1})$ by definition.

Next, we show that $\hat{F}(t_{L_2})$ represents a lower bound of $F(t_{L_2})$ by showing that $F(t_{L_2})$ FSD $\hat{F}(t_{L_2})$. Applying (3), we have that $\hat{\Pr}(\hat{t}_{L_1}) = \sum_{t_{L_1} \in m(\hat{t}_{L_1})} \hat{\Pr}(t_{L_1}) = \sum_{t_{L_1} \in m(\hat{t}_{L_1})} \Pr(t_{L_1})$. We show that $F(t_{L_2})$ FSD $\hat{F}(t_{L_2})$ as follows.

$$\begin{aligned} &\hat{F}(t_{L_2}) \\ &= \sum_{r \in V(\hat{T}_{L_1})} \hat{F}(t_{L_2}|\hat{T}_{L_1} = r)\hat{\Pr}(\hat{T}_{L_1} = r) \\ &= \sum_{r \in V(\hat{T}_{L_1})} [\max_{s \in m(r)} F(t_{L_2}|T_{L_1} = s) \sum_{t \in m(r)} \Pr(T_{L_1} = t)] \end{aligned} \tag{7}$$

Since every state of T_{L_1} must belong to a state of $\hat{T}_{L_1}$, we can expand the last equality in (7), recollect the terms, and show that $\hat{F}(t_{L_2}) \geq F(t_{L_2})$ as follows.

$$
\begin{aligned}
\hat{F}(t_{L_2}) &= \sum_{t_{L_1} \in V(T_{L_1})} [\max_{t_{L_1} \in m(\hat{t}_{L_1})} F(t_{L_2}|t_{L_1})] \Pr(t_{L_1}) \\
&\geq \sum_{t_{L_1} \in V(T_{L_1})} F(t_{L_2}|t_{L_1}) \Pr(t_{L_1}) \qquad (8) \\
&= F(t_{L_2})
\end{aligned}
$$

Therefore, we have shown that $F(t_{L_2})$ FSD $\hat{F}(t_{L_2})$, and that $\hat{F}(t_{L_2})$ represents a lower bound of $F(t_{L_2})$.

Now we assume that $F(t_{L_j})$ FSD $\hat{F}(t_{L_j})$ for j, and we prove that $F(t_{L_{j+1}})$ FSD $\hat{F}(t_{L_{j+1}})$ for j as follows. Using a similar procedure that we use to derive the inequality in (8) from the beginning of (7), we can derive the first inequality in (9), which is shown below.

$$
\begin{aligned}
\hat{F}(t_{L_{j+1}}) &\geq \sum_{t_{L_j} \in V(T_{L_j})} F(t_{L_{j+1}}|t_{L_j}) \hat{\Pr}(t_{L_j}) \qquad (9) \\
&= \sum_{t_{L_j} \in V(T_{L_j})} F(t_{L_{j+1}}|t_{L_j}) d\hat{F}(t_{L_j}) \\
&\geq \sum_{t_{L_j} \in V(T_{L_j})} F(t_{L_{j+1}}|t_{L_j}) dF(t_{L_j}) \\
&= F(t_{L_{j+1}})
\end{aligned}
$$

The second inequality in (9) follows from (2) based on the facts that $F(t_{L_j})$ FSD $\hat{F}(t_{L_j})$ for j and that link travel times have stochastic consistency property. Recall that, as we discuss in Section 3.1, $F(t_{L_{j+1}}|t_{L_j})$ of all links are non-increasing functions of t_{L_j} in a stochastically consistent network. Therefore, we have proved that $F(t_{L_{j+1}}) \leq \hat{F}(t_{L_{j+1}})$ for any $t_{L_{j+1}}$, so $F(t_{L_{j+1}})$ FSD $\hat{F}(t_{L_{j+1}})$.

Putting the previous results together, we have shown that, for all k, $F(t_{L_k})$ FSD $\hat{F}(t_{L_k})$ and $\hat{F}(t_{L_k})$ represents a lower bound of $F(t_{L_k})$.

3.4 Selecting states to aggregate

So far, we have shown that we can compute bounds on distributions over travel times by aggregating consecutive states. We have not mentioned how exactly we decide to cluster the original states. Assume that T have six states and that we want to aggregate these six states into two new states. Should we aggregate the first three states into a new state and the remaining states into another, or should we aggregate the first two states into a new state and the remaining states into another? How does this decision affect the tightness of bounds? Moreover, is it possible that we do not obtain bounds if we do not aggregate the original states appropriately?

The answer to the last question is that we will *always* obtain bounds as long as we aggregate the original, consecutive states. In the proof presented in Section 3.3, we do not assume how the consecutive states are aggregated, so the proof remains valid for answering this question.

The answers to the other questions are not clear to us. It is easy to make up numeric examples to show that the way we aggregate original states will affect the tightness of bounds. Experience indicates that the best clustering depends on the link travel times of those links that have *not* been added to the path. However, appending these "future" links to the path is not an acceptable option under our consideration. The computational costs would be too high.

Currently, we employ heuristics in selecting states to aggregate. Assume that we have determined to aggregate the original states of T into β states. We rank the original states according to $\hat{\Pr}(t)$, and mark the β states with the largest probability values. We then construct β new states, each with one such marked state and all unmarked states that are less than the marked state down to the next marked state. We add the remaining original states, if any, into the new state that contains the marked state with the largest value among all marked states.

3.5 Refining the aggregated states

Assume that we recover the distinctions among the consecutive states that are aggregated. Would the bounds become tighter or not? For instance, what would happen to the bounds if we refine the state space of T, say from $[t_0, t_1, t_2]$ and $[t_3, t_4, t_5]$ to $[t_0, t_1]$, $[t_2]$, and $[t_3, t_4, t_5]$?

Intuitively, we would expect that the bounds will tighten, since we would use more information in the computation. Indeed, we can show that the bounds of all travel times *always* tighten as we refine the aggregated states by introducing more distinctions between original states [11].

4 Application: path planning

In this section, we apply the state-space aggregation methods to an algorithm to find shortest paths from O to D. The algorithm attempts to find the best solution, under the constraint that we may not compute exact distributions over the travel times for comparing the paths. We then discuss features and extensions of the algorithm.

4.1 The algorithm

The algorithm maintains a priority queue, PQ, that contains partially expanded paths from O toward D. We prioritize paths by bounds on their *projected arrival times* at D. A projected arrival time at D of a path $O \to \cdots \to X$ is the random variable $\Pi = T_X + h(X)$, where $h(X)$ is the minimal amount of time required to travel from X to D. We can estimate $h(X)$ with the straight-line distance between X and D and the maximal driving speed from X to D, $\overline{v}_{XD}$. We can also estimate $h(X)$ with the shortest path distance from X to D and $\overline{v}_{XD}$.

In addition, we use the *lower envelope* of the projected arrival times of all paths in PQ to determine whether the algorithm should terminate. Let $LB_i(t)$ be the lower bound of the projected arrival time at D of path P_i in PQ. The lower envelope $LE(t)$ of the projected arrival times of all paths in PQ is defined as:

$$LE(t) = \max_{P_i \in PQ} LB_i(t).$$

Maintaining the lower envelope is easy. When new paths are added to PQ, we compare the current envelope with the lower bounds of the newly added paths, and update the lower envelope when necessary. When a path P_i is being removed, we only need to update $LE(t)$ for those ts such that $LE(t) = LB_i(t)$ before the removal of P_i. We describe the algorithm in the following elements:

Input: a stochastic network, the origin O with departure time, the destination D, and the maximal number of states β used to specify arrival times at intermediate locations after state-space aggregation.

Output: a selected path from O to D along with the maximum deviation of the distribution over its arrival time from the the distribution over the arrival time of the actual shortest path.

Data structures:

- PQ, a priority queue: Items associate a path with bounds on its path cost, with path costs prioritized by the method explained below.
- Lower envelope: The lower bound of all paths in PQ as is explained.
- Solution list (*SL*): Undominated candidate paths that connect O and D.

Procedure:

1. Add $O \to O$ to PQ with cost 0.
2. If PQ is empty and the solution list is empty, report an error. If PQ is empty and the solution list is not empty, terminate and return the best candidate in the solution list.
3. Remove the highest-priority path $P = O \to \cdots \to X$ from PQ, and update the lower envelope. If $X \neq D$, go to the next step. Otherwise, add *P* into the solution list unless there is a candidate solution with dominant priority in the solution list. Go to step 5.
4. Construct new paths for each possible next link the traveler could take from X, and insert the resulting items into PQ. Update the lower envelope.
5. If the lower envelope dominates the upper bound of a known path in the solution list, return the best candidate in the solution list. Otherwise, go to step 3.

Notice that it could be that neither path dominates the other when we compare two paths. Two paths might not be comparable in terms of first-order stochastic dominance even if we have exact distributions over arrival times. This could happen when the curves of the CDFs of their arrival times intersect.

We employ heuristics for prioritizing paths in PQ and the solution list. We can choose arbitrary measure that is consistent with the dominance ordering. For instance, we may prioritize paths based on expected values of their lower bounds.

The procedure can terminate when the lower envelope of projected arrival times of paths in PQ dominates the upper bound of a known path in the solution list. When this condition does occur, the known path must be better than all the paths in PQ since the projected arrival times are the earliest arrival time achievable by traveling on all paths. We show this proposition next.

4.2 Features of the algorithm

We show that, when the procedure terminates, the solution list must include the actual shortest path. Then, based on this result, we discuss how to compute maximum deviation of the distribution over the arrival time of the recommended path from the distribution over the arrival time of the actual shortest path.

We show the first proposition by contradiction. First, notice that all partial paths departing from O must reside in the priority queue or in the solution list. Let *Q* be the known path found in step 5, $AQ(t)$ denote the actual distribution over the arrival time at D of *Q*, and $UQ(t)$ the upper bound of $AQ(t)$. Denote the lower envelope of all paths in PQ by $LE(t)$. The following figure shows a generic picture of the scenario. If the actual shortest path *R* is not in the solution list and is still in PQ, then the actual distribution of *R* must fall into the lower-right hand side to the curve of $LE(t)$. This implies that the actual arrival time at D of *R* is an upper bound of that of *Q*. However, if the actual arrival time of *R* is an upper bound of that of *Q*, *Q* will be better than *R*, and *R* cannot be a shortest path. Therefore, we have shown that, if *R* is a shortest path, it must be in the solution list when the procedure terminates.

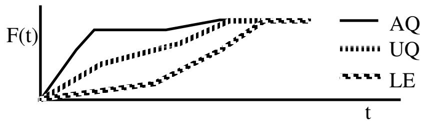

Figure 5: The actual shortest path must be in the solution list when the algorithm terminates.

As a result, when the procedure terminates, we can compute the maximum deviation of the distribution over the arrival time of the recommended path, *RP*, from the distribution over the arrival time of the actual shortest path. Let $LBS_i(t)$ and $UBS_i(t)$ be the lower and upper bounds on path P_i in the solution list, respectively. Let $LESL(t)$ and $UESL(t)$ be the lower and upper envelopes of all paths in the solution list, respectively. Namely, we define

$$\begin{aligned} LESL(t) &= \max_{P_i \in SL} LBS_i(t) \text{ and} \\ UESL(t) &= \min_{P_i \in SL} UBS_i(t). \end{aligned}$$

Since the exact distribution over the arrival time of the actual shortest path must fall in between $LESL(t)$ and $UESL(t)$, these envelopes can be used to compute the maximum deviation. For instance, assume that we choose to use the expected arrival time at D as the measure for the deviation. We can compute the expected arrival time calculated with $LESL(t)$, and report the difference between this expected arrival time and that of *RP* as the maximum deviation.

4.3 Anytime extensions

We may apply the algorithm in the context in which the deadlines for returning recommended paths are unknown at the time the algorithm is implemented. We may apply this algorithm in an iterative manner, setting β to 2 initially. If we can finish the procedure, we will have a candidate solution at our disposal. We may then set β to a larger quantity, say 3, and run the algorithm again to obtain another candidate solution. We can repeat the algorithm until the deadlines are reached, and return the candidate solution found in the latest completed run of the algorithm. If the deadlines are long enough, we may choose to terminate the computation if the candidate solution is good enough. This can happen, for example, when the maximum deviation of the candidate solution from the actual shortest path is considered small enough.

Several factors influence the variation of the computation time of the algorithm from iterations to iterations. On one hand, the increase of β tends to increase the costs for computing travel times of individual paths; on the other hand, we may reduce the number of candidate paths as we gather more information about path costs in each iteration. As we have proved in Section 4.2, the solution list must contain the actual shortest path when the procedure terminates at step 5. Therefore, in the subsequent run of the procedure with a larger β, we do not need to consider any path that cannot be expanded into paths that are in the solution list. Also, as we use more states in the computation when we increase β, the bounds will become tighter than their counterparts in previous runs of the algorithm. This makes non-optimal paths less likely to survive. In other words, the number of alternative paths that may be considered as candidate solutions should decrease as we increase β, thereby helping to keep the computational costs down.

This extended algorithm can demonstrate the desirable *anytime* property as discussed in [3]. Our algorithm can be interrupted and return a solution anytime after we finish the first run of the algorithm. Also, the quality of the recommended path will improve monotonically as we finish more iterations, if we define the quality as the maximum deviation between *LESL*(t) and *UESL*(t). However, the actual deviation between the recommended and the shortest path is *not* guaranteed to dwindle. There are always chances that the algorithm recommends the actual shortest path in early iterations and sub-optimal paths in subsequent iterations.

5 Conclusions

We report aggregation methods for computing bounds on travel times in stochastic networks. Our methods are complementary to the techniques presented by Bean and his colleague [2]. Their methods aggregate nodes and links into *macronodes* and *macroarcs* in transportation networks. Our methods aggregate the states of random variables and rely on the stochastic-dominance relationships among link travel times. We show that our methods always compute bounds of interest as long as we aggregate consecutive states. In addition, we manipulate the direction of bounds by applying either (4) or (5), and we control the tightness of bounds by refining the state space in the state-space aggregation operations. We also demonstrate applications of the bounds on travel times to shortest-path problems, and extend our algorithm so that performance of the overall system demonstrates the anytime property.

The findings reported in this paper rely on the stochastic consistency property of the transportation network. When not all link travel times have the stochastic consistency property, the reported methods may remain applicable in some special cases. We have found aggregation methods for computing bounds on travel times in these special cases, which we will report at a later date.

Acknowledgments

The first author would like to thank the Artificial Intelligence Laboratory of the University of Michigan and the Department of Computer Science of National Taiwan Ocean University for their support to this study.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[2] J. C. Bean, J. R. Birge, and R. L. Smith. Aggregation in dynamic programming. *Operations Research*, 35(2):215–220, 1987.

[3] M. Boddy and T. L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285, 1994.

[4] K.-C. Chang and R. Fung. Refinement and coarsening of Bayesian networks. In *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, pp. 475–482, Boston, Massachusetts, USA, 1990.

[5] N. Deo and C. Pang. Shortest path algorithms: Taxonomy and annotation. *Networks*, 14:275–323, 1984.

[6] E. W. Dijkstra. A note on two problems in connextion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[7] S. E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17:395–412, 1969.

[8] P. C. Fishburn and R. G. Vickson. Theoretical foundations of stochastic dominance. In Whitmore and Findlay [13], pp. 39–113.

[9] R. W. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182–188, 1986.

[10] D. E. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems applications. *IVHS Journal*, 1(1):1–11, 1993.

[11] C.-L. Liu and M. P. Wellman. Using qualitative relationships for bounding probability distributions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 346–353, Madison, Wisconsin, USA, 1998.

[12] M. P. Wellman, M. Ford, and K. Larson. Path planning under time-dependent uncertainty. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 532–539, Montréal, Québec, Canada, 1995.

[13] G. A. Whitmore and M. C. Findlay, editors. *Stochastic Dominance: An Approach to Decision Making Under Risk*. D. C. Heath and Company, 1978.