

Preprocessing Stochastic Shortest Path Problems on Directed Acyclic Networks

Daniel Reich

Program in Applied Mathematics, The University of Arizona, Tucson, AZ 85721-0089, dreich@math.arizona.edu,
<http://math.arizona.edu/~dreich/>

Leo Lopes

Department of Systems & Industrial Engineering, The University of Arizona, Tucson, AZ 85721-0020, leo@sie.arizona.edu,
<http://www.sie.arizona.edu/faculty/leolopes/>

We present an algorithm for preprocessing a class of stochastic Shortest Path Problems on directed acyclic networks. Our method significantly increases the utility of many existing frameworks. Given random costs with finite lower and upper bounds on each edge, our algorithm removes edges that cannot be in any optimal solution to the deterministic Shortest Path Problem, for any realization of the random costs. Although this problem is NP-complete, our algorithm efficiently preprocesses many edges in a given network and our computational results show that on average only .2% of the edges remain unclassified after preprocessing.

Key words: Preprocessing, Stochastic Optimization, Shortest Path, Directed Acyclic Network, Dynamic Programming, Integer Programming

History: This paper was first submitted on April 15, 2009.

1. Introduction

We present an algorithm for preprocessing a class of stochastic Shortest Path Problems on directed acyclic networks. Our aim is to reduce problem sizes by removing edges that cannot be in any optimal solution to the deterministic Shortest Path Problem for any realization of the random costs. Identifying which edges should be preprocessed has been classified as an NP-complete problem even when a network is restricted to be directed, acyclic and planar (Chanas and Zieliński 2003, Kasperski and Zieliński 2006). Given random costs with finite upper and lower bounds on each edge, the algorithm we present in this paper provides an efficient method for preprocessing almost all edges that are never in an optimal path.

In the past few decades, stochastic extensions of the deterministic Shortest Path Problem have received significant attention (Zieliński 2004, Yu and Yang 1998, Bard and Bennett 1991, Reich

and Lopes 2008), due to a multitude of applications where there is uncertainty. Several well known applications are transportation, where traffic conditions and weather are variable (Eiger et al. 1985); data transfer, where the number of bandwidth requests at a given time is unknown (Miller-Hooks 2001); emergency evacuation systems, where the size and dispersion of the population to be evacuated at the time of an emergency may be uncertain (Karbowicz and Smith 1984); and PERT applications where the completion times of activities in a project are not fixed (Burt and Garman 1971). Several stochastic extensions exist to satisfy the differing requirements of varying applications.

While dynamic programming approaches prove to be efficient for solving the deterministic Shortest Path Problem, the property that optimal paths consist of optimal subpaths is either missing or insufficient in many stochastic extensions of this problem. In the absence of dynamic programming approaches, an exponential number of paths combined with stochasticity results in significant computational challenges.

In robust optimization frameworks (Yu and Yang 1998, Zieliński 2004, Montemanni et al. 2004), the optimal path is characterized by the superiority of its worst-case performance to that of the other paths in a given network. Yu and Yang (1998) proved that the Robust Shortest Path Problem with a discrete scenario set is NP-hard, if the number of scenarios is bounded, and strongly NP-hard if the number of scenarios is unbounded. Kasperski and Zieliński (2006) showed that with a continuous scenario set, the problem remains NP-hard even on the class of series-parallel networks.

Perhaps the most utilized technique for finding optimal paths through stochastic networks is to maximize the expected values of utility functions. Bard and Bennett (1991) consider the case where completion time of activities is random and propose a heuristic to reduce the size of a stochastic network, in order to reduce the computational burden.

Aside from expected value problems, the other main branch of research in non-deterministic Shortest Path Problems focuses on the use of probability measures. Frank (1969) identified an exact method for obtaining the distribution function of the minimum cost path through a network

with random edge costs. His solution requires the use of multivariate integration, which in general can be computationally burdensome, if not intractable.

Sigal et al. (1980) consider the probabilities of paths being a solution to the deterministic Shortest Path Problem. In order to reduce the path dimension of their joint probability function, Sigal et al. (1979) introduce a cutset approach for dividing a network into independent and dependent path sets. However, evaluating the joint probability function of the remaining dependent path sets remains a non-trivial and burdensome computation. Reich and Lopes (2008) find the path with highest probability of being a solution the Shortest Path Problem efficiently, but their method is restricted to series-parallel networks.

Preprocessing edges significantly increases the utility of many existing frameworks. To preprocess, it is necessary to compare lower bounds on one set of edges with upper bounds on another set of edges. Finding which sets to compare is why this problem is NP-hard. Karasan (2001) use network layers to extract the edge sets that are compared and introduce a polynomial time method for preprocessing certain edges. Bard and Bennett (1991) use a node based approach to identify path comparisons of interest. Our approach has two phases. In the first phase, we use subnetworks to isolate edge set comparisons locally. In the second phase, we introduce a novel approach that uses integer programming. We aggregate all the local information from the first phase and propagate it to the entire network. Our computational results provide evidence that the use of global information significantly increases the number of edges that can be preprocessed efficiently.

The remainder of this paper is organized as follows. In Section 2, we provide background graph-theoretic concepts and define the class of stochastic Shortest Path Problems for which our preprocessing method applies. Section 3 presents our deterministic algorithm for preprocessing directed acyclic networks. Section 4 summarizes our computational results. Finally, Section 5 presents consequences of our work and future research directions.

2. Stochastic Shortest Path Problem Class

Let $G_{st} = (V_{st}, E_{st})$ be a directed acyclic network with *source* s and *sink* t , where V_{st} and E_{st} are its node and edge sets, respectively. For any nodes $q, r \in V_{st}$, let $G_{qr} = (V_{qr}, E_{qr})$ be the subnetwork

of G_{st} induced by all $q-r$ paths. Let K_{qr} be the set of all $q-r$ paths through G_{qr} . For every edge $e \in E_{st}$, let $[\underline{c}_e, \bar{c}_e]$ with $\underline{c}_e \geq 0$ be the range of possible costs for edge e . A *scenario* $\omega \in \Omega$ provides a realization of costs $c_e^\omega \in [\underline{c}_e, \bar{c}_e]$, $e \in E$; and the cost of any given set of edges E under scenario ω is given by

$$c^\omega(E) = \sum_{e \in k} c_e^\omega.$$

Throughout the paper we will use the notation for a given path to also represent the set of edges in that path.

DEFINITION 1 (STOCHASTIC SHORTEST PATH PROBLEM CLASS (SSPPC)). Consider a network G_{st} where the cost of each path $k \in K_{st}$ is defined as $\Phi_k = \sum_{e \in k} C_e$, with C_e taking values on $[\underline{c}_e, \bar{c}_e]$. The *Stochastic Shortest Path Problem Class* is the class of problems whose objective is only affected by $s-t$ paths that have minimum cost for some realization of the random variables. All the problems in Section 1 or special cases thereof are instances of the SSPPC class.

Although the random edge costs in G_{st} are independent of one another, the random path costs are not, since many edges may be present in multiple paths. Consequently, solving SSPPC problems is difficult on general networks. Since there are an exponential number of paths, enumerating all paths would be computationally intractable, even if the measure for each path could be easily computed. The alternative is to use Monte Carlo methods, but even these are intractable for large networks. Our preprocessing method aims to reduce this computational time on general directed acyclic networks by eliminating unnecessary parts of the network deterministically.

3. SSPPC Preprocessing Algorithm

The preprocessing problem that our algorithm addresses can be formalized with the following standard definitions from the literature (Karasan (2001), Kasperski and Zieliński (2006)).

DEFINITION 2 (WEAK AND NON-WEAK EDGES). Let SP_{qr}^ω be an optimal path for the Shortest Path Problem on G_{qr} under scenario ω . We define an edge e as *weak* on G_{qr} if there exists an ω such that e is in SP_{qr}^ω . Otherwise, we refer to e as a *non-weak* edge on G_{qr} .

In any SSPPC framework, edges that are non-weak will neither be part of an optimal path nor will have any effect on an optimal solution. Hence, preprocessing these non-weak edges can improve running-time on many computationally difficult problems.

Preprocessing edges is difficult due to the existence of an exponential number of edge cost combinations. The key idea that our SSPPC Preprocessing Algorithm leverages is that certain cost combinations can provide significant insight; our aim is to isolate those combinations.

To detect if an edge e is non-weak, our strategy is to identify competition for e and then allow e to hold its greatest cost advantage against the competition. This is non-trivial since edges that share paths with e may also be in paths that provide competition to e . Definitions 3 and 4 provide a foundation for isolating e against its competition in various subnetworks G_{qr} of G_{st} .

DEFINITION 3 ($q-r$ SUBNETWORK UPPER BOUND PATH). A $q-r$ subnetwork upper bound path \overline{SP}_{qr} with cost U_{qr} is an optimal path for the Shortest Path Problem on G_{qr} with upper bounds costs \bar{c} on all edges in E_{qr} .

Our next step is to establish conditional lower bounds for edge $e = (u, v)$, given a subnetwork upper bound \overline{SP}_{qr} .

DEFINITION 4 ($q-r$ SUBNETWORK LOWER BOUND PATH). A $q-r$ subnetwork lower bound path $\underline{SP}_{qr}|\overline{SP}_{qr}$ with cost $L_{qr}|\overline{SP}_{qr}$ is formed by joining at edge e optimal solutions to Shortest Path Problems on G_{qu} and G_{vr} , where all edges in \overline{SP}_{qr} have upper bound costs and all other edges in $E_{qr} \setminus \overline{SP}_{qr}$ have lower bound costs. In other words, $\underline{SP}_{qr}|\overline{SP}_{qr}$ is a $q-r$ subpath containing e with least possible cost, given upper bound costs on all edges in \overline{SP}_{qr} .

Later in this section, we introduce and prove the mathematics behind the SSPPC Preprocessing Algorithm and formalize the algorithm itself, but first let us provide the intuition for our method with the example in Figure 1. Steps (b) - (f) leverage the subnetwork upper bound and lower bound paths defined above. The method for step (g) will be introduced in Subsection 3.2.

3.1. Dominated Subpaths

We justify step (e) in Figure 1 by showing that if a subnetwork lower bound path $\underline{SP}_{qr}|\overline{SP}_{qr}$ has higher cost than its corresponding upper bound path \overline{SP}_{qr} , then e is non-weak on G_{qr} .

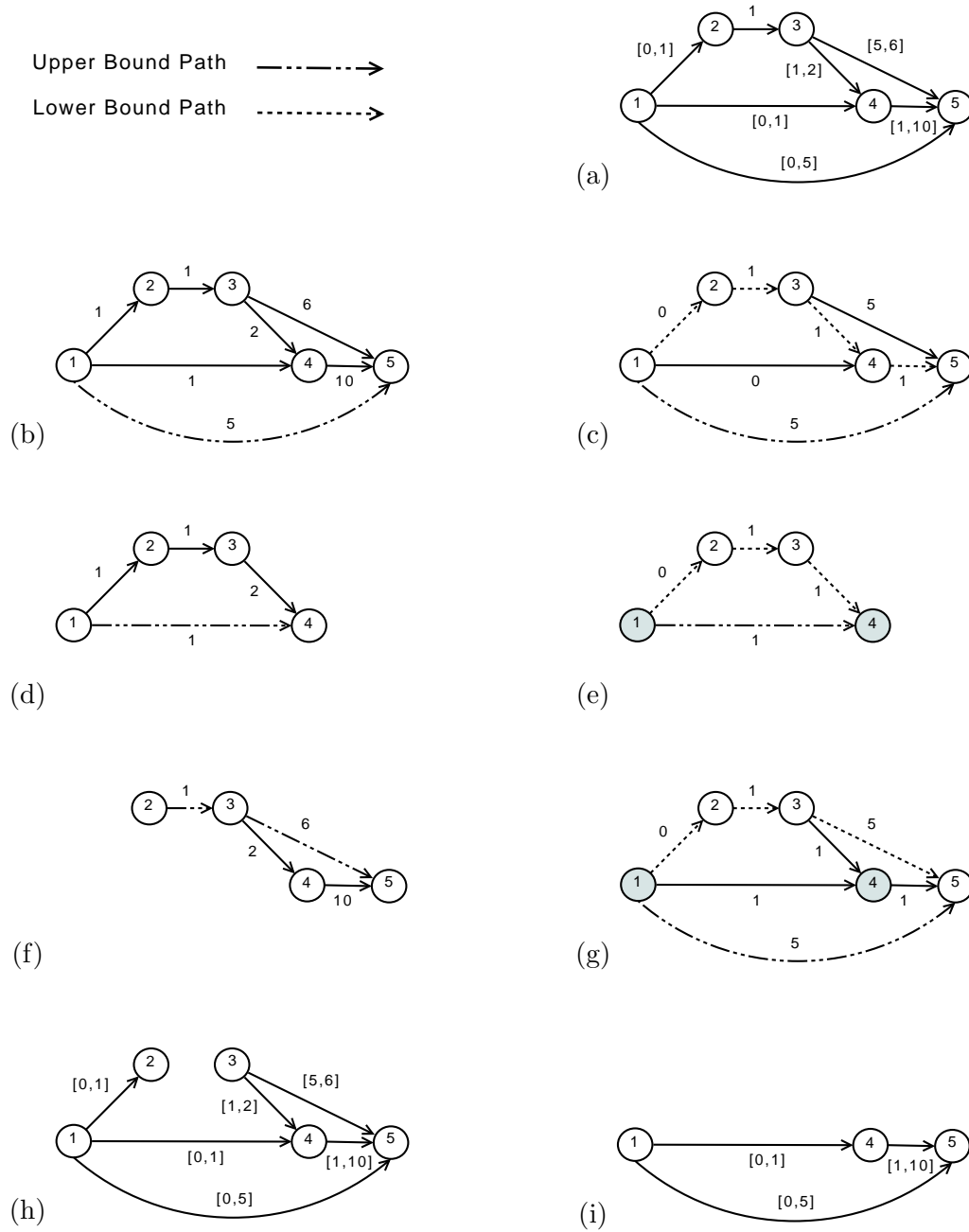


Figure 1 The SSPPC Preprocessing Algorithm illustrated for preprocessing edge (2,3). (a) The original network with cost intervals on each edge. (b) $\overline{SP}_{1,5}$. (c) $\underline{SP}_{1,5} | \overline{SP}_{1,5}$. (d) $\overline{SP}_{1,4}$. (e) $\underline{SP}_{1,4} | \overline{SP}_{1,4}$. All paths from node 1 to node 4 are eliminated (as indicated by gray shading) since $U_{1,4} < L_{1,4} | \overline{SP}_{1,4}$. (f) $\overline{SP}_{2,5}$. Since edge (2,3) is part of $\overline{SP}_{2,5}$, there is no need to find $L_{2,5} | \overline{SP}_{2,5}$. (g) The least costly path through edge (2,3) between node 1 and node 5 that does not pass through both nodes 1 and 4, with lower bound costs on all edges except those in the upper bound found in (b). (h) Edge (2,3) is removed since the cost of the lower bound path in (g) is less than the cost of the upper bound path in (b). (i) The network after preprocessing is complete.

LEMMA 1. If $L_{qr}|\overline{SP}_{qr} > U_{qr}$ then $c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) > c^\omega(SP_{qr}^\omega)$ for any $\omega \in \Omega$.

Proof of Lemma 1. Let E_{qr}^ω be the set of edges that are included in paths $(SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega)$ and \overline{SP}_{qr} , then

$$\begin{aligned} & c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + \bar{c}(E_{qr}^\omega) - c^\omega(E_{qr}^\omega) \\ &= c^\omega(SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega \setminus E_{qr}^\omega) + \bar{c}(E_{qr}^\omega) \\ &\geq \underline{c}(SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega \setminus E_{qr}^\omega) + \bar{c}(E_{qr}^\omega) \\ &\geq L_{qr}|\overline{SP}_{qr}. \end{aligned}$$

Consequently,

$$\begin{aligned} & c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) \\ &\geq L_{qr}|\overline{SP}_{qr} + c^\omega(E_{qr}^\omega) - \bar{c}(E_{qr}^\omega) \\ &> U_{qr} + c^\omega(E_{qr}^\omega) - \bar{c}(E_{qr}^\omega) \\ &\geq SP_{qr}^\omega. \end{aligned}$$

□

Note: The converse of Lemma 1 is not true, as can be seen in Figure 2, where

$$L_{qr}|\overline{SP}_{qr} = 19 < 20 = U_{qr}$$

but for all ω ,

$$c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) > c^\omega(SP_{qr}^\omega)$$

since

$$\underline{c}_e + c^\omega(SP_{vr}^\omega) > c^\omega(SP_{ur}^\omega).$$

We justify step (g) in Figure 1 by Corollary 1 below. If a subnetwork lower bound path $\underline{SP}_{qr}|\overline{SP}_{qr}$ has higher cost than its corresponding upper bound path \overline{SP}_{qr} , then no path containing e and

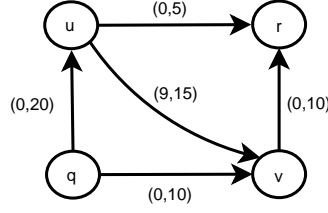


Figure 2 This shows that the converse of Lemma 1 is not true since $L_{qr}|\overline{SP}_{qr} = 19 < 20 = U_{qr}$.

passing through nodes q and r can be optimal for the Shortest Path Problem on G_{st} given any realization of the random edge costs.

COROLLARY 1. *If $L_{qr}|\overline{SP}_{qr} > U_{qr}$, then for every ω , $c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + c^\omega(SP_{rt}^\omega) > c^\omega(SP_{st}^\omega)$.*

Proof of Corollary 1. By Lemma 1, if $L_{qr}|\overline{SP}_{qr} > U_{qr}$ then $c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) > c^\omega(SP_{qr}^\omega)$. Consequently, for all ω ,

$$\begin{aligned}
 & c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + c^\omega(SP_{rt}^\omega) \\
 & > c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qr}^\omega) + c^\omega(SP_{rt}^\omega) \\
 & \geq c^\omega(SP_{st}^\omega).
 \end{aligned}$$

□

Lemma 1 and Corollary 1 isolate subnetworks in order to isolate competition for edge e . Another alternative for isolating competition through edge e is to consider the entire network G_{st} while still focusing on various subnetworks G_{qr} . Definition 5 and Lemma 2 formalize this idea.

DEFINITION 5 ($q-r$ GLOBAL LOWER BOUND PATH). A $q-r$ global lower bound path $\underline{SP}_{sqr}|\overline{SP}_{st}$ with cost $L_{sqr}|\overline{SP}_{st}$ is formed by joining at edge e optimal solutions to Shortest Path Problems on G_{sq} and G_{qu} with optimal solutions on G_{vr} and G_{rt} , where all edges in \overline{SP}_{st} have upper bound costs and all other edges in $E_{st} \setminus \overline{SP}_{st}$ have lower bound costs. In other words, $\underline{SP}_{sqr}|\overline{SP}_{st}$ is an $s-t$ path containing both a $q-r$ subpath and edge e with least possible cost, given upper bound costs on all edges in \overline{SP}_{st} .

We show in Lemma 2 below that if a global lower bound path $\underline{SP}_{sqr}|\overline{SP}_{st}$ has higher cost than its corresponding upper bound path \overline{SP}_{st} , then no path containing e and passing through nodes q and r can be optimal for the Shortest Path Problem on G_{st} given any realization of the random edge costs.

LEMMA 2. *If $L_{sqr}|\overline{SP}_{st} > U_{st}$, then for every ω , $c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + c^\omega(SP_{rt}^\omega) > c^\omega(SP_{st}^\omega)$.*

Proof of Lemma 2. Let E_{st}^ω be the set of edges that are included in paths $(SP_{sq}^\omega \cup SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega \cup SP_{rt}^\omega \setminus E_{st}^\omega)$ and \overline{SP}_{st} , then

$$\begin{aligned} & c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + c^\omega(SP_{rt}^\omega) + \bar{c}(E_{st}^\omega) - c^\omega(E_{st}^\omega) \\ &= c^\omega(SP_{sq}^\omega \cup SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega \cup SP_{rt}^\omega \setminus E_{st}^\omega) + \bar{c}(E_{st}^\omega) \\ &\geq \underline{c}(SP_{sq}^\omega \cup SP_{qu}^\omega \cup \{e\} \cup SP_{vr}^\omega \cup SP_{rt}^\omega \setminus E_{st}^\omega) + \bar{c}(E_{st}^\omega) \\ &\geq L_{sqr}|\overline{SP}_{st}. \end{aligned}$$

Consequently,

$$\begin{aligned} & c^\omega(SP_{sq}^\omega) + c^\omega(SP_{qu}^\omega) + \underline{c}_e + c^\omega(SP_{vr}^\omega) + c^\omega(SP_{rt}^\omega) \\ &\geq L_{sqr}|\overline{SP}_{st} + c^\omega(E_{st}^\omega) - \bar{c}(E_{st}^\omega) \\ &> U_{st} + c^\omega(E_{st}^\omega) - \bar{c}(E_{st}^\omega) \\ &\geq SP_{st}^\omega. \end{aligned}$$

□

Lemma 1, Corollary 1 and Lemma 2 provide conditions for restricting certain subpaths containing e in the given network G_{st} . The following definition characterizes these restrictions.

DEFINITION 6 (DOMINATED). Edge e is *dominated locally* in subnetwork G_{qr} if $L_{qr}|\overline{SP}_{qr} > U_{qr}$. Edge e is *dominated globally* in subnetwork G_{qr} if $L_{sqr}|\overline{SP}_{st} > U_{st}$. Edge e is *dominated* in

subnetwork G_{qr} if either it is dominated locally in G_{qr} or it is dominated globally in subnetwork G_{qr} .

3.2. Subpath Constrained Shortest Path Problem

Our next step is to simultaneously enforce all the restrictions where edge e is dominated; specifically we must prevent e from using paths which violate the restrictions. In order to do so, we introduce an integer program which we refer to as the Subpath Constrained Shortest Path Problem.

We wish to solve the Shortest Path Problem on G_{st} , while ensuring the following two constraints.
 e in Optimal Path: The shortest path includes a specified edge $e \in E$.

Dominated Subpath Constraints: The shortest path does not contain any dominated $q - r$ subpath.

We can formulate the *e in optimal path* constraint as

$$x_e = 1.$$

On directed acyclic networks, there exists a partial ordering, so if $e = (u, v)$ and $q \preceq u \preceq v \preceq r$, then all $q - r$ subpaths can be eliminated. Otherwise, no $q - r$ subpath could possibly contain edge e , so no $q - r$ subpaths need be eliminated. Let $D = \{(q, r) \in V \times V : e \text{ is dominated}\}$. Then the *dominated subpath constraints* can be formulated as

$$\sum_{j:(q,j) \in E} x_{qj} + \sum_{j:(j,r) \in E} x_{jr} \leq 1$$

for all $(q, r) \in D$.

3.2.1. Integer Programming Formulation Let c_{ij} be the cost on edge $(i, j) \in E$, where all edges in \overline{SP}_{st} have upper bound costs and all other edges in $E_{st} \setminus \overline{SP}_{st}$ have lower bound costs. Let x_{ij} be a $\{0, 1\}$ decision variable, where $x_{ij} = 1$ iff edge (i, j) is included in the optimal path and $x_{ij} = 0$ otherwise. Given a set of dominated subpaths D and the edge $e = (u, v) \in E$ currently being preprocessed, the following formulation finds the shortest $s - t$ path r^* containing e such that no subpath in D is a subpath of r^* .

$$\text{Minimize} \quad \sum_{(i,j) \in E} c_{ij} x_{ij} \quad (1)$$

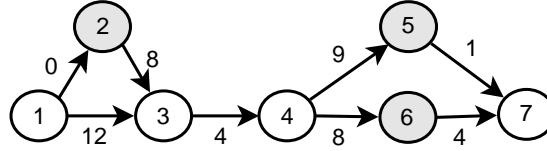


Figure 3 This Subpath Constrained Shortest Path Problem is not solved by its LP relaxation. The optimal IP solution path with $e = (3, 4)$ and $D = \{(2, 5), (2, 6)\}$ is $1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7$ and has cost 26. In the LP relaxation flow is split between $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7$ and $1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7$ resulting in a lower cost of 25.

subject to

$$\sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{for all } i \notin \{s, t\} \\ -1 & \text{if } i = t \end{cases} \quad (2)$$

$$\sum_{j:(q,j) \in E} x_{qj} + \sum_{j:(j,r) \in E} x_{jr} \leq 1 \quad \forall (q, r) \in D \quad (3)$$

$$x_{uv} = 1 \quad (4)$$

$$x_{ij} = \{0, 1\} \quad \text{for all } (i, j) \in E \quad (5)$$

The objective is to minimize the cost function (1) subject to the flow balance constraints (2), the *dominated subpath constraints* (3) and the *e in path constraint* (4). Since G_{st} is a directed acyclic network, no negative-cost *cycle* exists. Consequently, the objective ensures that there is at most one edge entering or leaving any node in the shortest path, as the objective is to minimize cost.

Although the constraint matrix for the Shortest Path Problem is totally unimodular and the linear programming relaxation provides an integer solution, in general the same is not true for the Subpath Constrained Shortest Path Problem. Figure 3 provides an example where the linear programming relaxation provides a fractional solution. In practice the solution is often obtained by the linear programming relaxation, as is shown in our computational results in Section 4.

If e is weak, then it will often be the case that the optimal solution to the IP provides a cost realization to verify weakness. Specifically, one can check whether the optimal path to the IP with all its edge costs at their lower bounds and all other edge costs at their upper bounds also solves the Shortest Path Problem with those costs. Our computational results in Section 4 will provide evidence that checking this condition is a highly effective method for verifying that edges are weak.

Theorem 1 guarantees that if there is no feasible solution to the Subpath Constrained Shortest Path Problem, then edge e is non-weak. Theorem 2 guarantees that if the optimal solution to the IP has a cost greater than U_{st} then edge e is also non-weak.

THEOREM 1. *Let $D \subseteq \{V \times V\}$ be the set of all subnetworks on which edge e is dominated. Let K'_{st} be the set of all $s-t$ paths through G that contain edge e . Let $K'_{sqr} \subseteq K'_{st}$ be the set of all $s-t$ paths through G that contain edge e and pass through nodes q and r . Let $K^*_{st} \subseteq K'_{st}$ satisfy that $\forall (q, r) \in D, k \notin K'_{sqr}$. If $K^*_{st} = \emptyset$, then e is non-weak.*

Proof of Theorem 1. Assume $K^*_{st} = \emptyset$. Then $\forall k \in K'_{st}, \exists (q, r) \in D$ such that $k \in K'_{sqr}$. However, by Corollary 1 and Lemma 2, if $k \in K'_{sqr}$ for $(q, r) \in D$, then k is suboptimal for all ω . Thus e is non-weak. \square

THEOREM 2. *Consider edge costs c' where all edges in \overline{SP}_{st} have upper bound costs \bar{c} on and all other edges in $E_{st} \setminus \overline{SP}_{st}$ have lower bound costs \underline{c} . If $K^*_{st} \neq \emptyset$ and if $\forall k \in K^*_{st}, c'(k) > U_{st}$, then e is non-weak.*

Proof of Theorem 2. Assume $K^*_{st} \neq \emptyset$ and that $\forall k \in K^*_{st}, c'(k) > U_{st}$. For every $k \in K^*_{st}$ and $\forall \omega$,

$$\begin{aligned}
& c^\omega(k) + \bar{c}(k \cap \overline{SP}_{st}) - c^\omega(k \cap \overline{SP}_{st}) \\
&= c^\omega(k \setminus \overline{SP}_{st}) + \bar{c}(k \cap \overline{SP}_{st}) \\
&\geq \underline{c}(k \setminus \overline{SP}_{st}) + \bar{c}(k \cap \overline{SP}_{st}) \\
&= c'(k) \\
&> U_{st}.
\end{aligned}$$

Therefore

$$c^\omega(k) > U_{st} - \bar{c}(k \cap \overline{SP}_{st}) + c^\omega(k \cap \overline{SP}_{st}) \geq c^\omega(\overline{SP}_{st}).$$

Consequently, $\forall \omega$ the cost of every path $k \in K^*_{st}$ is higher than the cost of path \overline{SP}_{st} . Additionally, by Theorem 1 for every path $k \in K'_{st} \setminus K^*_{st}$, k is suboptimal for all ω . Thus e is non-weak. \square

Algorithm 1 The SSPPC Preprocessing Algorithm takes a directed acyclic network G_{st} and preprocess out any non-weak edges that are detected.

Find $s - t$ upper bound path \overline{SP}_{st} and its cost U_{st}

for all edges $e = (u, v) \in E_{st} \setminus \overline{SP}_{st}$ **do**

for all $(q, r) \in V \times V$ such that $q \preceq u \preceq v \preceq r$ **do**

if Edge e is dominated in subnetwork G_{qr} **then**

$D = D \cup \{(q, r)\}$

end if

end for

Run Subpath Constrained Shortest Path Problem on G_{st} with edge e and subpath elimination set D

if Optimal solution has cost greater than U_{st} or no solution exists **then**

 Remove e from E_{st}

else

 Set the cost of all edges in the optimal solution to their lower bounds

 Set the cost of all other edges to their upper bounds

if Optimal solution also solves the Shortest Path Problem **then**

 Edge e is weak

else

 Edge e is not classified as weak or non-weak

end if

end if

end for

Algorithm 1 summarizes our SSPPC Preprocessing Algorithm. We have established correctness for this algorithm in Theorems 1 and 2. After preprocessing, in any given network there may be a small subset of edges that cannot be classified as either being weak or being non-weak. To

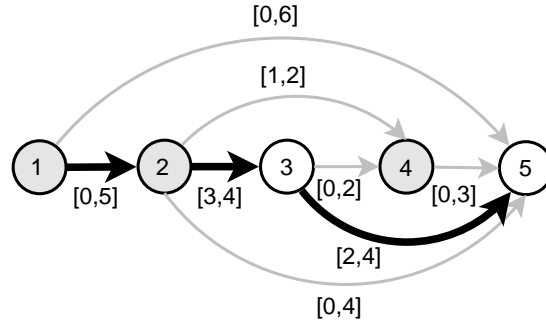


Figure 4 This figure provides an example where the optimal solution (in bold) to the Subpath Constrained Shortest Path Problem fails to detect that $e = (2, 3)$ is non-weak, where $D = \{(1, 4), (2, 4)\}$.

illustrate where our method fails to detect a non-weak edge, consider the network in Figure 4. In the next section we provide computational results that demonstrate the effectiveness of the SSPPC Preprocessing Algorithm in preprocessing networks by removing non-weak edges and verifying weak edges.

4. Computational Results

We tested our SSPPC Preprocessing Algorithm on directed acyclic networks with 677, 790, 903, 1016 and 1128 edges, all of which have 48 nodes. For each network size, we tested fifty networks. These computations were performed on identical machines with Intel Core 2 CPUs @2.4 GHz and 4GB RAM.

We designed our experiment to test the algorithm's effectiveness at processing networks with a significant number of non-weak edges. While our algorithm does not guarantee that all edges can be classified as weak or non-weak, in our computational results on average only .2% of the edges were unclassified after preprocessing. Moreover, out of all 250 networks tested, at most 2.2% of the edges in any network remained unclassified after the SSPPC Preprocessing Algorithm terminated.

We use Perl modules to generate random test networks. First, we build *complete* directed acyclic networks with 48 nodes where there exists an edge from node i to node j whenever $i < j$. For the 1128 edge networks tested no edges are removed. For the 1016, 903, 790 and 677 edge networks, 10%, 20%, 30% and 40% of the edges, respectively, were removed. In order to create random topologies, an edge is selected randomly. If removing that edge does not result in disconnectivity

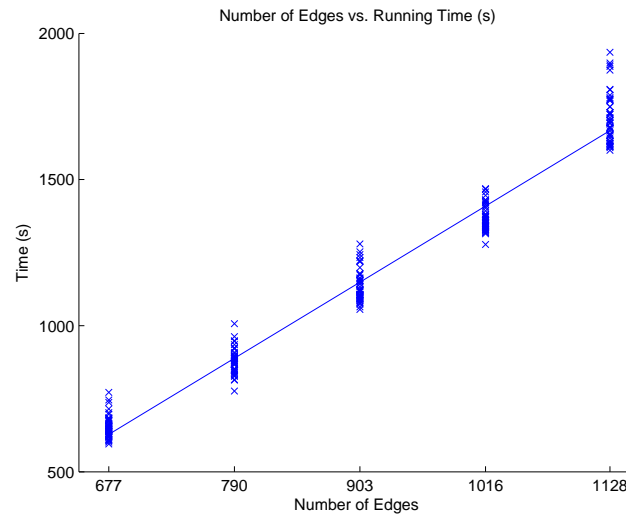


Figure 5 The running time distribution of 50 tests networks for each edge size is shown above. The linear trend shows that the running time is increasing at an approximately linear rate in the number of edges for a fixed number of nodes ($|V| = 48$).

between the 48 nodes, then that edge is removed. Otherwise another edge is selected at random. This process is repeated until the desired number of edges remain. The inputs to the module are the number of nodes and the percentage of edges to be removed.

A second Perl module then reads in the topology and randomly assigns lower and upper bound costs to each edge in the network. The costs lower bounds are assigned via uniform random variables distributed between 0 and 48; the cost upper bounds are then assigned via uniform random variables distributed between the corresponding lower bound and 48.

We implemented SSPPC using a combination of Perl and AMPL. In the Perl module, the dominated subpaths are identified. The information for the Subpath Constrained Shortest Path Problem is then sent to an AMPL model, which is solved using CPLEX 11.2. The integer constraint is then relaxed to test if the linear relaxation provides an integer solution. The edge is then identified as weak, non-weak or unclassified.

Out of the 39854 Subpath Constrained Shortest Path Problems that were run, the linear programming relaxation did not provide an optimal integer solution in only 36 cases. The running time of the algorithm was dominated by the $O(|V|^2)$ implementation of Dijkstra's Algorithm as

Stats	# Edges	Time	# of IPs	# of IPs not solved by LP relaxation
Mean	677	653.02	91.82	0.26
StDev	0	37.04	82.6	1.56
Min	677	595.22	0	0
Max	677	771.8	344	11
Mean	790	877.58	119.94	0.02
StDev	0	41.24	85.52	0.14
Min	790	776.19	3	0
Max	790	1006.56	332	1
Mean	903	1135.46	180.18	0.16
StDev	0	53.13	113.87	0.65
Min	903	1055.67	3	0
Max	903	1279.83	386	4
Mean	1016	1367.76	157.18	0.1
StDev	0	43.46	98.69	0.58
Min	1016	1277.99	0	0
Max	1016	1468.94	391	4
Mean	1128	1708.46	247.96	0.18
StDev	0	85.97	143.15	0.63
Min	1128	1600.08	0	0
Max	1128	1935.57	523	4

Table 1 This table shows the computational preprocessing running times and number of IPs needed for each of the five network sizes tested. For each size, 50 networks were tested and the mean, standard deviation, min and max for each column is listed. The rightmost column shows that nearly all of the IPs solved with their LP relaxation.

can be evidenced in the linear trend in Figure 5. Further detail about the running time, the number of IPs and the number of IPs solved by their linear relaxation is summarized in Table 1.

Due to the randomness of the topologies and the randomness of the costs, there was significant variation in the number of weak and non-weak edges in the networks tested. On average, 13% of the edges were identified as weak but the range was from 32% to just a single edge. Accordingly, on average 87% of the edges were identified by the SSPPC Preprocessing Algorithm as non-weak and were therefore removed. The Subpath Constrained Shortest Path Problem was needed for 18% of the edges, whereas 82% of the edges were removed as a result of being locally dominated on the

Stats	# Edges	# After Pre-processing	# Removed w/o IP	# Removed w/ IP	# Proved Weak w/o IP	# Proved Weak w/ IP	# In UBst	# Not Classified
Mean	677	67.28	582.78	26.94	0.72	63.94	1.68	0.94
StDev	0	52.25	83.26	35.31	0.97	50.18	0.71	2.32
Min	677	2	328	0	0	0	1	0
Max	677	226	675	130	3	221	3	13
Mean	790	92.52	667.32	30.16	0.84	88.72	1.9	1.06
StDev	0	54.98	85.97	34.55	0.93	53.29	0.65	2.52
Min	790	4	454	0	0	3	1	0
Max	790	212	786	134	3	205	3	15
Mean	903	128.72	719.94	54.34	0.92	123.56	1.96	2.28
StDev	0	68.62	114.26	53.87	0.9	66.33	0.78	4.24
Min	903	4	515	0	0	3	1	0
Max	903	282	899	218	4	278	4	20
Mean	1016	119.5	856.06	40.44	0.9	115.22	1.86	1.52
StDev	0	68.46	99.04	37.66	0.86	66.82	0.86	2.26
Min	1016	1	622	0	0	0	1	0
Max	1016	299	1015	140	3	290	4	10
Mean	1128	178.3	876.9	72.8	1.2	171.36	1.94	3.8
StDev	0	91.66	144.15	61.33	1.11	87.94	0.79	5.13
Min	1128	1	599	0	0	0	1	0
Max	1128	359	1127	216	4	341	4	22

Table 2 This table shows the computational preprocessing results for each of the five network sizes tested. For each size, 50 networks were tested and the mean, standard deviation, min and max for each column is listed. The rightmost column shows that nearly all edges are identified as being weak or being non-weak. The other columns show where these identifications took place within the algorithm.

G_{st} network. Further detail about the computational results for classification of edges is provided in Table 2.

5. Summary and Conclusions

For a given directed acyclic network, with random edge costs, we have developed a computationally efficient method for identifying non-weak edges in practice. In doing so, we have introduced an algorithm that utilizes Dijkstra’s Algorithm to obtain information that is then sent to an integer programming problem, namely the Subpath Constrained Shortest Path Problem. Rather than using traditional modeling approaches to develop a pure integer programming formulation to solve this

NP-complete problem, we utilized integer programming in formulating a relatively easy to solve subproblem. While we face the trade-off that a small number of edges remain unclassified after the SSPPC Preprocessing Algorithm terminates, we have shown that our method is highly efficient in practice.

The SSPPC Preprocessing Algorithm provides added utility to many complex stochastic Shortest Path optimization frameworks by efficiently removing non-weak edges. For many practical stochastic shortest path applications, the data sets that are used to support analysis via stochastic optimization frameworks may contain significant noise, resulting in computational burdens. In practice, the SSPPC Preprocessing Algorithm can be implemented so that the edges in any given network are preprocessed in parallel.

In future research, the methodology developed in this paper can be applied to more general stochastic network flow and stochastic optimization problems. Additionally, for specific frameworks within a class of stochastic optimization problems there may be potential to develop more specialized preprocessing methods.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. DMS-0602173.

References

- Bard, J. F., J. E. Bennett. 1991. Arc reduction and path preference in stochastic acyclic networks. *Management Science* **37** 198–215.
- Burt, J. M., Jr., M. B. Garman. 1971. Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science* **18** 207–217.
- Chanas, S., P. Zieliński. 2003. On the hardness of evaluating criticality of activities in a planar network with duration intervals. *Operations Research Letters* **31** 53–59.
- Eiger, A., P. B. Mirchandani, H. Soroush. 1985. Path preferences and optimal paths in probabilistic networks. *Transportation Science* **19** 75–84.

- Frank, H. 1969. Shortest paths in probabilistic graphs. *Operations Research* **17** 583–599.
- Karasan, MC and Yaman, O.E. and Pinar. 2001. The robust shortest path problem with interval data. Tech. rep., Bilkent University, Department of Industrial Engineering.
- Karbowicz, C.J., J.M. Smith. 1984. A k-shortest paths routing heuristic for stochastic network evacuation models. *Engineering Optimization* **7** 253–280.
- Kasperski, A., P. Zieliński. 2006. The robust shortest path problem in series–parallel multidigraphs with interval data. *Operations Research Letters* **34** 69–76.
- Miller-Hooks, E. 2001. Adaptive least-expected time paths in stochastic, time-varying transportation and data networks. *Networks* **37** 35–52.
- Montemanni, R., L. M. Gambardella, A. V. Donati. 2004. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters* **32** 225–232.
- Reich, D., L. Lopes. 2008. The most likely path. *Submitted to Networks* .
- Sigal, C. E., A. A. B. Pritsker, J. J. Solberg. 1979. The use of cutsets in Monte Carlo analysis of stochastic networks. *Mathematics and Computers in Simulation* **21** 376–384.
- Sigal, C. E., A. A. B. Pritsker, J. J. Solberg. 1980. The stochastic shortest route problem. *Operations Research* **28** 1122–1129.
- Yu, G., J. Yang. 1998. On the robust shortest path problem. *Computers and Operations Research* **25** 457–468.
- Zieliński, P. 2004. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research* **158** 570–576.