

The Most Likely Path

Daniel Reich

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

Leo Lopes

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

In this paper, we present a stochastic shortest path problem that we refer to as the *Most Likely Path Problem* (MLPP). We prove that optimal solutions to the MLPP are composed of optimal subpaths, a property which is essential for solving the classical deterministic shortest path problem. On series-parallel networks, we produce analytical bounds for the probability of the Most Likely Path (MLP), which we compute efficiently via dynamic programming and ordinal optimization.

Key words: Stochastic Optimization, Shortest Path, Series-Parallel, Probability Bounds

History: This paper was first submitted on September 12, 2008.

1. Introduction

In this paper, we present a stochastic shortest path problem that we refer to as the *Most Likely Path Problem* (MLPP). We attempt to answer the following question: given a network topology and distributions on the costs of each edge, is there a path that is significantly more likely to cost less than all the other paths? The answer to this question is often yes. In these cases, the MLPP provides unique insights into the networks.

Our interest in the MLPP is motivated by contexts where identifying an optimal path in a network provides broader and more useful information than can be obtained from a single edge. Additionally, in many contexts for shortest path problems with stochastic costs, one does not have the benefit of repeated instances of the problem. As a consequence, identifying a *Most Likely Path* (MLP) may provide more valuable information than would be obtained by identifying an expected shortest path.

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 [12, 20]. In the absence of dynamic programming approaches, exponential numbers of paths result in computational difficulties and limit the application of some of the stochastic methods that have been developed.

Recently, robust optimization frameworks have emerged for modeling shortest paths involving stochasticity [23, 24, 17]. In these frameworks, the optimal path is characterized by the superiority of its worst-case performance to that of the other paths in the network. Yu and Yang [23] 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 [14] showed that with a continuous scenario set, the problem remains NP-hard on the class of series-parallel networks.

Perhaps the most used technique for finding optimal paths through stochastic networks is to maximize the expected value of a utility function. Loui [16] uses this technique to extend the von

Neumann-Morgenstern [see 10] formulation for identifying ideal methods of evaluation in the presence of uncertainty. Bard and Miller [3] solved constrained probabilistic shortest path problems by using utility functions to weight dynamic programming solutions to deterministic networks. Utility functions that vary inversely with path cost have also been used to provide dynamic programming approaches to stochastic problems by Bard and Bennett [2] and Eiger et al. [9].

Aside from expected value problems, the other main branch of research in non-deterministic shortest path problems focuses on the use of a probability measure, *i.e.* distribution functions and joint probability functions. For a given network, many edges may be present in multiple paths, resulting in dependent random variables and therefore complicated probability measures. Frank [12] 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. Consequently, he provides a sampling based method for statistically analyzing the distribution functions. Burt and Garman [5] develop an alternate sampling method that uses conditional Monte Carlo simulation for estimating distribution functions along paths in stochastic networks. Adlakha [1] combines ideas from the cutset approach of Sigal et al. [19] with aspects of the simulation method of Burt and Garman [5] to compute distribution functions of the minimum cost paths in stochastic networks.

Sigal et al. [20] define optimality indices corresponding to all paths in a given stochastic network as the probabilities of paths being a solution to the deterministic shortest path problem. While the formulation of Frank [12] aims to identify the distribution of the minimum cost path, the formulation of Sigal et al. [20] aims to compare and rank all paths within a given network. The latter objective also requires evaluating joint probability functions, which is non-trivial due to dependence of random path cost variables.

In order to reduce the path dimension of their joint probability function, Sigal et al. [20] 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. Their method of solution is theoretically precise, and their paper

is frequently cited, but rarely extended, due to computational difficulties involving multivariate integration and combinatorial numbers of paths.

We define a solution to the MLPP as a path with greatest probability of realizing the least cost, which stated in terms of Sigal et al. [20] is a path with highest optimality index. As opposed to previous computationally intractable methods for finding optimality indices on general networks, our interest is in identifying the most general class of networks for which the problem is tractable.

We show that on the class of series-parallel networks, many complications due to dependence of random variables can be resolved via dynamic programming. Our approach introduces a Monte Carlo sampling heuristic based on ordinal optimization, and in doing so, avoids the high computational cost of multivariate integration. Valdes et al. [21] present a linear-time algorithm for recognizing series-parallel networks. Series-parallel networks have been studied extensively in electrical engineering and have been applied to network flow problems [7, 15, 18, 22, 4].

The remainder of this paper is organized as follows. In Section 2, we provide background graph-theoretic concepts, formulate the MLPP and prove subpath optimality for the MLPP on general networks. Section 3 provides a formal definition of series-parallel networks and introduces our new subclass of *essential series-parallel networks*. Section 4 presents our computationally tractable algorithm for identifying the MLP and bounding its probability on series-parallel networks. Section 5 summarizes our computational results. Finally, Section 6 presents consequences of our work and future research directions.

2. Formulation of the Most Likely Path Problem

Given a network topology and its corresponding edge cost (or edge length) vector, the deterministic shortest path problem can be solved easily as either a dynamic programming problem or as a linear programming problem. However in many situations, portions of the network are unobservable and the cost vector is therefore unknown. The MLPP is intended to analyze cases where at least part, if not all, of a given cost vector is composed of independent random variables with probability density functions. For realizations of these random variables, solutions to the deterministic shortest

path problem can be obtained. We wish to find a solution that would be obtained most frequently under repeated realizations, *i.e.* a path with greatest probability of being shortest.

In this paper we use the notation in Cook et al. [8] whenever possible, augmenting it when necessary for dealing with series-parallel networks. Let $G_{st} = (V_{st}, E_{st})$ be a directed network with *source* s and *sink* t , where V_{st} and E_{st} are its vertex and edge sets, respectively. For any nodes $u, v \in V_{st}$, let $G_{uv} = (V_{uv}, E_{uv})$ be the subnetwork of G_{st} induced by all $u - v$ paths. Let K_{uv} be the set of all $u - v$ paths through G_{uv} .

DEFINITION 1 (MOST LIKELY PATH PROBLEM). Consider a network G_{st} where at least part if not all of the edge cost vector C is composed of continuous random variables. We model all fixed costs as degenerate random variables with Dirac-Delta probability density functions, which are centered at their respective fixed costs. (Note: Dirac-Delta functions are not Lebesgue integrable and are therefore not probability density functions in the strict sense of the terminology. However, throughout the remainder of this paper, we treat them as probability density functions in order to avoid burdensome notation, as they satisfy all required properties for any proofs that follow.) Let f_C denote the vector of probability density functions corresponding to C . Let the cost of each path $i \in K_{st}$ be defined as $\Phi_i = \sum_{e \in i} C_e$.

The *Most Likely Path Problem* is the problem of finding an $s - t$ path (not necessarily unique) with highest probability of being the shortest path, *i.e.*, a path r_{st}^* satisfying

$$r_{st}^* \in \arg \max_{i \in K_{st}} P \left(\Phi_i = \min_{k \in K_{st}} \Phi_k \right), \quad (1)$$

where

$$P \left(\Phi_i = \min_{k \in K_{st}} \Phi_k \right) = P \left(\bigcap_{k \in K_{st} \setminus \{i\}} \{ \Phi_i \leq \Phi_k \} \right). \quad (2)$$

We refer to r_{st}^* as the *Most Likely Path* (MLP).

Although the random edge costs in C are independent of one another, the random path costs are not, since edges may appear in multiple paths. Consequently, solving the MLPP is difficult on general networks; known solution methods require evaluating the joint probability function in (1),

which in turn requires multivariate integration. Moreover, since there are an exponential number of paths, solving for the objective in (1) by enumerating all paths is computationally intractable, even if the joint probability function is easily computed. An alternative is to use a Monte Carlo method, but even these are intractable for large networks. Our method reduces this computational time by orders of magnitude by producing bounds on the probability of the MLP.

2.1. Redundant Fixed Costs and Degeneracy

In the MLPP, fixed costs represent certainty on the part of the modeler about parts of a given network, whereas variable costs represent uncertainty. We demonstrate that from a modeling perspective, in our stochastic network model, equivalent fixed cost paths should be preprocessed out of any given network, so that only one remains.

When multiple equivalent fixed cost paths remain in any given network, if any of these equivalent fixed cost paths has non-zero probability of being the MLP, then

$$\sum_{i \in K_{st}} P \left(\Phi_i = \min_{k \in K_{st}} \Phi_k \right) > 1 \quad (3)$$

and we refer to the network as *degenerate*. In non-degenerate networks, these probabilities must always sum to one, *i.e.*,

$$\sum_{i \in K_{st}} P \left(\Phi_i = \min_{k \in K_{st}} \Phi_k \right) = 1,$$

since: (i) continuous random variables (random costs) are not equal to one another (almost surely); (ii) continuous random variables (random costs) are not equal to fixed costs (almost surely); and (iii) fixed path costs are either not equal to one another or are never optimal. Consequently, in these non-degenerate networks, the interpretation of each path probability is intuitive. We now demonstrate why (3) holds in degenerate networks; and why these networks should be avoided from a modeling perspective.

Consider a degenerate network with three independent paths. Let two of the three paths have identical fixed costs of 5 and the third have a random cost uniformly distributed on $[0,10]$. The comparison that one random variable is less than or equal to another, which seems natural, measures

all three paths as having probability $\frac{1}{2}$ of being the shortest path. The reason is that when either fixed cost path is a shortest path, the other fixed cost path is also a shortest path. Consequently, the probabilities of the three paths being shortest sum to 1.5 and all three appear to be equally desirable. If one of the fixed cost paths were removed, the two remaining paths would still be shortest with probability $\frac{1}{2}$, and therefore would still be equally desirable. Since the removed path is clearly equivalent to the remaining fixed cost path, that too would be equally desirable. So the redundant fixed cost path could be preprocessed out of the network without losing any information.

Alternatively, consider a non-degenerate network with three independent paths. Let two of the three paths have independent random costs uniformly distributed on $[5 - \epsilon, 5 + \epsilon]$ and the third have a random cost uniformly distributed on $[0, 10]$. The third path is shortest whenever it has cost between 0 and $5 - \epsilon$, which occurs with probability $\sim \frac{1}{2}$. One of the other two paths is shortest whenever the third has cost between $5 + \epsilon$ and 10, which also occurs with probability $\sim \frac{1}{2}$. But since those two paths now have continuous random costs, the probability that they are equal is 0; and therefore both are shortest with probability $\sim \frac{1}{4}$. The probabilities of these three paths being shortest sum to 1; and all three are now crucial for identifying the MLP.

2.2. Subpath Optimality for the MLPP

In deterministic shortest path problems, subpath optimality allows for dynamic methods of solution. We show that subpath optimality holds for the MLPP as well.

THEOREM 1. *Let all random edge costs on G_{st} be independent random variables. Let r_{st}^* pass through nodes $u \in V_{st}$ and $v \in V_{st}$. Then if r_{uv}^* is a subpath of r_{st}^* , then r_{uv}^* is an optimal solution to the MLPP on G_{uv} .*

Proof of Theorem 1. We show that if there exists a path \tilde{r}_{uv} such that

$$P\left(\Phi_{\tilde{r}_{uv}} = \min_{k \in K_{uv}} \Phi_k\right) > P\left(\Phi_{r_{uv}^*} = \min_{k \in K_{uv}} \Phi_k\right) \quad (4)$$

then path r_{uv}^* is not a subpath of r_{st}^* . Since we are concerned with optimality on G_{uv} , without loss of generality, we can consider a restricted network from s to t composed of all paths in G_{st} that

pass through nodes u and v . Partition r_{st}^* into three subpaths: $s - u$ path r_{su}^* ; $u - v$ path r_{uv}^* ; and $v - t$ path r_{vt}^* . Since edge costs are independent, we have

$$P\left(\Phi_{r_{st}^*} = \min_{k \in K_{st}} \Phi_k\right) = P\left(\Phi_{r_{su}^*} = \min_{k \in K_{su}} \Phi_k\right) P\left(\Phi_{r_{uv}^*} = \min_{k \in K_{uv}} \Phi_k\right) P\left(\Phi_{r_{vt}^*} = \min_{k \in K_{vt}} \Phi_k\right).$$

If there exists a path \tilde{r}_{uv} with

$$P\left(\Phi_{\tilde{r}_{uv}} = \min_{k \in K_{uv}} \Phi_k\right) > P\left(\Phi_{r_{uv}^*} = \min_{k \in K_{uv}} \Phi_k\right),$$

then there exists a path \tilde{r}_{st} with

$$P\left(\Phi_{\tilde{r}_{st}} = \min_{k \in K_{st}} \Phi_k\right) = P\left(\Phi_{r_{su}^*} = \min_{k \in K_{su}} \Phi_k\right) P\left(\Phi_{\tilde{r}_{uv}} = \min_{k \in K_{uv}} \Phi_k\right) P\left(\Phi_{r_{vt}^*} = \min_{k \in K_{vt}} \Phi_k\right).$$

Consequently, if equation (4) holds then \tilde{r}_{st} is more likely than r_{st}^* , which is a contradiction. Thus r_{uv}^* is optimal for G_{uv} \square .

We have just shown that subpath optimality holds for the MLPP on general networks. However, unlike in the deterministic case, subpath optimality alone is in general not sufficient for constructing a dynamic programming-based solution method for the MLPP. Specifically, the joint probability measure (1) cannot be computed iteratively, due to event dependence. We will see that in series-parallel networks, this event dependence can be handled iteratively.

3. Series-Parallel Networks

While subpath optimality holds for the MLPP on general networks, it is insufficient for constructing a dynamic programming-based solution method; however, when combined with specialized topological properties, these solution methods become possible. We show that on the class of series-parallel networks, complications due to dependence of random events can be resolved. In this section, we introduce the class of series-parallel networks and then define the new specialized class of *essential series-parallel networks*. These essential networks play a key role in our solution method for the MLPP on the class of series-parallel networks.

DEFINITION 2 (SERIES-PARALLEL NETWORKS [4]). A network G_{st} is called *series-parallel* (also known as two-terminal series-parallel, edge series-parallel, strongly series-parallel) if it can be constructed by combinations of the following three rules:

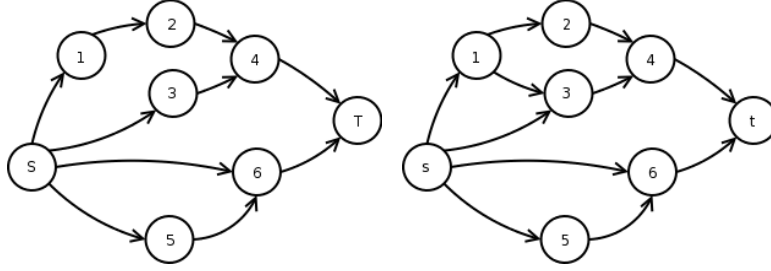


Figure 1 The network on the left is series-parallel. The one on the right is not, *i.e.* notice edge (1,3).

1. *Base Network*: Any network $G_{st}(V_{st}, E_{st})$ with $V_{st} = \{s, t\}$, $E_{st} = \{(s, t)\}$ is series-parallel.
2. *Parallel Composition*: Let $G_{s_1 t_1}(V_{s_1 t_1}, E_{s_1 t_1})$ and $G_{s_2 t_2}(V_{s_2 t_2}, E_{s_2 t_2})$ be two series-parallel networks with sources s_1 and s_2 and sinks t_1 and t_2 , respectively. By setting $s_p = s_1 = s_2$ and $t_p = t_1 = t_2$, we form the series-parallel network $G_{s_p t_p}$.
3. *Series Composition*: By setting $t_1 = s_2$ or $t_2 = s_1$, we form the series-parallel network $G_{s_1 t_2}$ or $G_{s_2 t_1}$, respectively.

Figure 1 illustrates series-parallel networks. The special structure of series-parallel networks allows for our dynamic programming method of solution to the MLPP on this class of networks.

As part of our algorithm for solving the MLPP on series-parallel networks, we dynamically compare the MLP with all other paths. We then obtain probability bounds for the MLP with a simplified series-parallel network topology, which we define as follows.

DEFINITION 3 (ESSENTIAL SERIES-PARALLEL NETWORKS). A network $\tilde{G}_{st}(\tilde{V}_{st}, \tilde{E}_{st})$ is *essential series-parallel* if:

1. There exists an ordering function $f: \tilde{V}_{st} \rightarrow \mathbb{N}$, such that $f(s) = 1$, $f(t) = |\tilde{V}_{st}|$, $f(s) < f(v) < f(t)$ for all $v \in \tilde{V}_{st} \setminus \{s, t\}$, and $f(v_i) \neq f(v_j)$ for all $v_i \neq v_j \in \tilde{V}_{st}$.
2. If $f(v_i) - f(v_j) = 1$, then there exists either one or two edges between nodes v_i and v_j for all $v_i, v_j \in \tilde{V}_{st}$.
3. If $f(v_i) - f(v_j) \neq 1$, then there exists at most one edge between v_i and v_j for all $v_i, v_j \in \tilde{V}_{st}$.
4. $\tilde{G}_{st}(\tilde{V}_{st}, \tilde{E}_{st})$ is series-parallel.

Figure 2 provides an example of an essential series-parallel network.

Algorithm 1 Reduce any series-parallel network to an essential series-parallel network.

Choose any $s - t$ path $k \in K_{st}$.

$$\tilde{V}_{st} = V_{st}.$$

$$\tilde{E}_{st} = E_{st}.$$

$$\text{Flag} = 0.$$

while Flag == 0 **do**

$$\text{Flag} = 1.$$

for all $(u, v) \in \tilde{E}_{st} \setminus k$ **do**

if there exists more than one edge $(u, v) \in \tilde{E}_{st} \setminus k$ **then**

Parallel Decomposition: Remove from \tilde{E}_{st} all but one edge $(u, v) \in \tilde{E}_{st} \setminus k$.

$$\text{Flag} = 0.$$

end if

end for

for all $(u, v) \in \tilde{E}_{st} \setminus k$ **do**

if $\exists q \in \tilde{V}_{st}$ such that $(q, u) \in \tilde{E}_{st} \setminus k$ **then**

if $\nexists q' \in \tilde{V}_{st}$ such that $(q', u) \in \tilde{E}_{st} \setminus (q, u)$ and $\nexists v' \in \tilde{V}_{st}$ such that $(u, v') \in \tilde{E}_{st} \setminus (u, v)$ **then**

Series Decomposition: $\tilde{E}_{st} = \{\tilde{E}_{st} \setminus \{(q, u), (u, v)\}\} \cup \{(q, v)\}$. $\tilde{V}_{st} = \tilde{V}_{st} \setminus \{u\}$.

$$\text{Flag} = 0.$$

end if

else if $\exists r \in \tilde{V}_{st}$ such that $(v, r) \in \tilde{E}_{st} \setminus k$ **then**

if $\nexists u' \in \tilde{V}_{st}$ such that $(u', v) \in \tilde{E}_{st} \setminus (u, v)$ and $\nexists r' \in \tilde{V}_{st}$ such that $(v, r') \in \tilde{E}_{st} \setminus (v, r)$ **then**

Series Decomposition: $\tilde{E}_{st} = \{\tilde{E}_{st} \setminus \{(u, v), (v, r)\}\} \cup \{(u, r)\}$. $\tilde{V}_{st} = \tilde{V}_{st} \setminus \{v\}$.

$$\text{Flag} = 0.$$

end if

end if

end for

end while

return \tilde{G}_{st}

THEOREM 2. *Any series-parallel network G_{st} can be reduced to an essential series-parallel network \tilde{G}_{st} via Algorithm 1.*

Proof of Theorem 2. We first prove that as result of the series decompositions, \tilde{V}_{st} consists only of nodes in V_{st} that are connected to edges in k . Assume this was not true, then there must exist some node $v \in \tilde{V}_{st}$ such that:

- (a) There does not exist any edge $(u, v) \in k$ for $u \in \tilde{V}_{st}$; and there does not exist any edge $(v, r) \in k$ for $r \in \tilde{V}_{st}$.
- (b) There exists at least one edge $(u', v) \in \tilde{E}_{st} \setminus k$ for $u' \in \tilde{V}_{st}$; and there exists at least one edge $(v, r') \in \tilde{E}_{st} \setminus k$ for $r' \in \tilde{V}_{st}$.

Let $E_{\cdot v}^{\text{in}}$ be the set of edges $\{(u', v) \in \tilde{E}_{st} \setminus k : u' \in \tilde{V}_{st}\}$. Since G_{st} is series-parallel, v must be the sink of some series-parallel network $G_{\cdot v}$ whose edge set contains $E_{\cdot v}^{\text{in}}$. Since the edges in k are consecutively connected, the sink of $G_{\cdot v}$ would have to be connected to one of the edges in k if $G_{\cdot v}$ contained any edge in k . Therefore, by (a) above, $G_{\cdot v}$ must not contain any edge in k . Consequently, by Definition 2 (series-parallel networks), $G_{\cdot v}$ can be reduced to a single edge via combinations of series and parallel decompositions.

Similarly, we can identify and reduce the series-parallel network G_v whose edge set contains $\{(v, r') \in \tilde{E}_{st} \setminus k : r' \in \tilde{V}_{st}\}$. Hence, without loss of generality we can consider the case (b) above when there is exactly one edge $(u', v) \in \tilde{E}_{st} \setminus k$ for some $u' \in \tilde{V}_{st}$ and one edge $(v, r') \in \tilde{E}_{st} \setminus k$ for some $r' \in \tilde{V}_{st}$. However, these two edges are replaced by a single edge during the series decomposition step and node v is removed, which is a contradiction to our assumption.

We have just shown that \tilde{V}_{st} consists only of nodes in V_{st} that are connected to edges in k . Let us now prove that the four components of Definition 3 (essential series-parallel networks) hold.

1. Order the nodes connecting edges in k according to the order of those edges: $f(s) = 1$, $f(v) = 2$ for $(s, v) \in k$, etc.

2. By 1, whenever $f(v_i) - f(v_j) = 1$ there exists an edge $(v_i, v_j) \in k$. By the parallel decomposition step, there cannot exist more than one other edge between v_i and v_j .

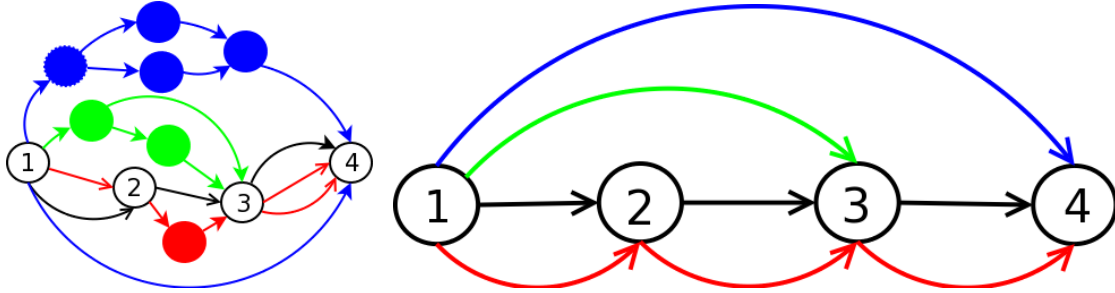


Figure 2 The network on the left is series-parallel. The one on the right is essentially series-parallel and is formed from the one on the left.

3. By 1, whenever $f(v_i) - f(v_j) \neq 1$ there exists no edge $(v_i, v_j) \in k$. By the parallel decomposition step, there cannot exist more than one edge between v_i and v_j .

4. Neither parallel decompositions nor series decompositions compromise the series-parallel topology of a network. Therefore the resulting network must be series-parallel.

Therefore any series-parallel network G_{st} can be reduced to an essential series-parallel network \tilde{G}_{st} via Algorithm 1 \square .

4. Algorithm for Solving the MLPP on Series-Parallel Networks

4.1. SPMLP: Identifying the MLP on Series-Parallel Networks

In this section, we introduce our algorithm for solving the MLPP on the class of series-parallel networks, which we refer to as SPMLP. In general, there is no known iterative process that can be used to evaluate the joint probability measure in our objective (1). As a consequence, multivariate integration is unavoidable, which is true even on general series-parallel networks. Fortunately, on series-parallel networks, it is not necessary to evaluate the joint probability in equation (2) in order to locate the optimal path. Instead, we can identify the MLP with a dynamic Monte Carlo sampling approach, which uses ordinal optimization to increase computational efficiency. While this approach locates the MLP, it does not compute the probability of the MLP. Alternatively, we provide analytical lower and upper bounds for the joint probability (1), which is computed efficiently and without the need for multivariate integration. SPMLP locates the MLP via dynamic Monte Carlo sampling and measures its probability via dynamically computed lower and upper bounds.

4.2. Dynamic Monte Carlo Sampling using Ordinal Optimization

Consider two subpaths r_{uv}^1 and r_{uv}^2 of any directed network G_{st} (not necessarily series-parallel), which originate at node u and terminate at node v . In order to eliminate either r_{uv}^1 or r_{uv}^2 as a potential subpath of the optimal path r_{st}^* , we need not precisely measure both sides of inequality (4); rather we need only identify which side is greater.

Ho et al. [13] introduce *ordinal optimization* as a means of comparing designs rather than estimating accurately the performance measures of those designs. They argue that ordinal optimization provides a basis for efficient preprocessing, which then reduces the computational burden of the solution step. In the context of the MLPP, precise measurements of both sides of inequality (4) could be obtained via extensive Monte Carlo sampling; however, by utilizing ordinal optimization, we are able to limit the sampling needed by aiming to identify only whether inequality (4) is true or false; not to obtain precise measurements of both sides.

For each sample, we run Dijkstra's algorithm on a realization of subnetwork G_{uv} and introduce indicator functions to keep track of the number of times r_{uv}^1 or r_{uv}^2 are the shortest paths. Note that these indicator functions are random variables whose expectations are the probabilities that r_{uv}^1 and r_{uv}^2 are the shortest paths within subnetwork G_{uv} . By only tracking one of the two indicator functions at each sample, we ensure that the indicator functions are independent from one another. We use the following statistical analysis to derive our stopping condition: Our null hypothesis is that the means of the two indicator functions are equal, but the variances are both unequal and unknown. After we reach 120 degrees of freedom, we compute a t -statistic at each iteration. Since the t distribution is essentially normal at this point, we use the percentile from the normal distribution (1.960) for this test. If we are able to establish the desired 95% confidence interval, we select the path whose sample mean is greater. If we are unable to establish this confidence interval within 10^4 iterations (2×10^4 samples), we stop since either (i) both subpaths will be eliminated at a later stage of the algorithm or (ii) both subpaths are virtually identical in probability.

As a consequence of Theorem 1, by eliminating one of the two subpaths, we in turn eliminate from the set of potential solutions all paths containing the eliminated subpath. Therefore when G_{st}

is series-parallel, we can identify the MLP with high accuracy in $O(|E|)$ steps. Next, we measure the probability that for a given realization of the network, this solution is a shortest path, *i.e.* we wish to measure (2) for this solution. However, as previously stated, this either requires multivariate integration or extensive sampling.

We instead use a highly efficient algorithm on a probabilistically equivalent essential series-parallel network to obtain lower and upper bounds for the desired probability. Before describing the method for obtaining the essential series-parallel network of interest, let us introduce the necessary probability background information.

4.2.1. Combining Edges in Parallel Let X and Y be independent random costs, with distribution functions F_X and F_Y , respectively. The distribution of the random variable $C := \min\{X, Y\}$ can be found as follows. Since for any c ,

$$P(C > c) = P(X > c)P(Y > c),$$

we have that

$$F_C(x) = 1 - ([1 - F_X(x)][1 - F_Y(x)]), \quad (5)$$

where F_C is the cumulative distribution of C .

4.2.2. Combining Edges in Series Let X and Y be independent random costs, with probability density functions f_X and f_Y . The distribution of the random variable $C := X + Y$ is given by

$$P(X + Y \leq c) = \int \int_{x+y \leq c} f_X(x)f_Y(y)dx dy = \int_{-\infty}^{\infty} F_X(c - y)f_Y(y)dy = F_X \star f_Y. \quad (6)$$

4.3. Essential Series-Parallel Networks

In order to compute the probability bounds for an MLP r_{st}^* , we make use of the simplified essential network topology. With respect to r_{st}^* , we are able to reduce the series-parallel network G_{st} to the simpler *essential series-parallel* topology \tilde{G}_{st} via Algorithm 1.

Through Algorithm 1, the *aggregate edge* set E' can be formed by iteratively collapsing all edges in parallel and series that are not edges in r_{st}^* . At each series decomposition, the cost distribution of the new edge is found by applying the convolution operator (6). At each parallel decomposition, the cost distribution of the remaining edge is found by applying the multiplication operator (5). By construction, the probability measure of interest is unchanged under both series and parallel decompositions.

4.4. Lower Bound on Essential Series-Parallel Networks

We say that two events A and B are *positively correlated* if $P(A \cap B) \geq P(A)P(B)$. Consider an essential series-parallel network \tilde{G}_{st} that was constructed with respect to path r_{st}^* . In this section, we show that all events corresponding to path r_{st}^* in \tilde{G}_{st} are positively correlated. We then use this correlation to separate our objective (2) on the essential network into multiple one-dimensional integrals, thereby producing a valid lower bound.

In order to show the needed inequality, *i.e.* positive correlation, we measure indicator functions of these events via the expected value, using the results from the following Theorem. The proof of Theorem 3 is analogous to the proof by Chayes et al. [6] of the Harris-FKG Inequality [11], which is commonly used in percolation theory.

THEOREM 3. *Let $\theta_n : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\phi_n : \mathbb{R}^n \rightarrow \mathbb{R}$ be non-increasing functions, *i.e.* for every i , $\theta_n(x_1, \dots, x_i, \dots, x_n) \leq \theta_n(x_1, \dots, \tilde{x}_i, \dots, x_n)$ if $x_i \geq \tilde{x}_i$ and $\phi_n(x_1, \dots, x_i, \dots, x_n) \leq \phi_n(x_1, \dots, \tilde{x}_i, \dots, x_n)$ if $x_i \geq \tilde{x}_i$. Let X_1, \dots, X_n be independent real-valued random variables with probability density functions f_1, f_2, \dots, f_n , respectively. Then*

$$E[\theta_n(X_1, X_2, \dots, X_n) \phi_n(X_1, X_2, \dots, X_n)] \geq E[\theta_n(X_1, X_2, \dots, X_n)] E[\phi_n(X_1, X_2, \dots, X_n)].$$

Proof of Theorem 3.

Base case $n = 1$:

Since both θ_1 and ϕ_1 are non-increasing, for any x_1, \tilde{x}_1 the quantities $\theta_1(x_1) - \theta_1(\tilde{x}_1)$ and $\phi_1(x_1) - \phi_1(\tilde{x}_1)$ are either both non-negative, or both negative. Thus:

$$\begin{aligned}
0 &\leq \int \int [\theta_1(x_1) - \theta_1(\tilde{x}_1)] [\phi_1(x_1) - \phi_1(\tilde{x}_1)] f_1(x_1) f_1(\tilde{x}_1) dx_1 d\tilde{x}_1 \\
&= \int \int \theta_1(x_1) \phi_1(x_1) f_1(x_1) f_1(\tilde{x}_1) dx_1 d\tilde{x}_1 - \int \int \theta_1(x_1) f_1(x_1) \phi_1(\tilde{x}_1) f_1(\tilde{x}_1) dx_1 d\tilde{x}_1 \\
&\quad - \int \int \theta_1(\tilde{x}_1) f_1(\tilde{x}_1) \phi_1(x_1) f_1(x_1) dx_1 d\tilde{x}_1 + \int \int \theta_1(\tilde{x}_1) \phi_1(\tilde{x}_1) f_1(\tilde{x}_1) f_1(x_1) dx_1 d\tilde{x}_1 \\
&= \int \theta_1(x_1) \phi_1(x_1) f_1(x_1) dx_1 \int f_1(\tilde{x}_1) d\tilde{x}_1 - \int \theta_1(x_1) f_1(x_1) dx_1 \int \phi_1(\tilde{x}_1) f_1(\tilde{x}_1) d\tilde{x}_1 \\
&\quad - \int \theta_1(\tilde{x}_1) f_1(\tilde{x}_1) d\tilde{x}_1 \int \phi_1(x_1) f_1(x_1) dx_1 + \int \theta_1(\tilde{x}_1) \phi_1(\tilde{x}_1) f_1(\tilde{x}_1) d\tilde{x}_1 \int f_1(x_1) dx_1 \\
&= E[\theta_1(X_1) \phi_1(X_1)] - E[\theta_1(X_1)] E[\phi_1(X_1)] - E[\theta_1(X_1)] E[\phi_1(X_1)] + E[\theta_1(X_1) \phi_1(X_1)] \\
&= 2E[\theta_1(X_1) \phi_1(X_1)] - 2E[\theta_1(X_1)] E[\phi_1(X_1)].
\end{aligned}$$

So,

$$E[\theta_1(X_1) \phi_1(X_1)] \geq E[\theta_1(X_1)] E[\phi_1(X_1)].$$

Induction hypothesis Assume the result is true for $n \leq k$:

Induction step $n = k + 1$:

The proof of the induction step is accomplished via the following four steps:

1. We condition on X_1, \dots, X_k to obtain an outer expectation over a random variable of dimension k and an inner expectation over a random variable of dimension 1.

2. Since the inner expectation is one-dimensional, *i.e.*, over the $k + 1$ -st dimension, we apply the result of the base case to obtain a product of one-dimensional expectations. Each of the two resulting one-dimensional expectations is a random variable of dimension k .

3. We apply the induction hypothesis to the product of random variables of dimension k .

4. We use the definition of conditional expectation to obtain the desired result.

Step 1: We condition on X_1, \dots, X_k to obtain an outer expectation over a random variable of dimension k and an inner expectation over a random variable of dimension 1.

$$\begin{aligned}
& E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) \phi_{k+1} (X_1, X_2, \dots, X_{k+1})] \\
&= E [E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) \phi_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]], \tag{7}
\end{aligned}$$

where the inner expectation,

$$\begin{aligned}
& E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) \phi_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k] \\
&= \int_{\mathbb{R}} \theta_{k+1} (X_1, X_2, \dots, X_k, x_{k+1}) \phi_{k+1} (X_1, X_2, \dots, X_k, x_{k+1}) f_{k+1} (x_{k+1}) dx_{k+1}. \tag{8}
\end{aligned}$$

Step 2: Since the inner expectation is one-dimensional, *i.e.*, over the $k+1$ -st dimension, we apply the result of the base case to obtain a product of one-dimensional expectations. Each of the two resulting one-dimensional expectations is a random variable of dimension k .

Since by assumption $\theta_{k+1}(x_1, x_2, \dots, x_{k+1})$ and $\phi_{k+1}(x_1, x_2, \dots, x_{k+1})$ are non-increasing functions of x_{k+1} , by the base case $n = 1$, we have

$$\begin{aligned}
& E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) \phi_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k] \\
&\geq E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k] E [\phi_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]. \tag{9}
\end{aligned}$$

Notice that the resulting product of one-dimensional expectations (9) is in fact a product of random variables of dimension k .

Step 3: We apply the induction hypothesis to the product of random variables of dimension k .

Note that $\int_{\mathbb{R}} \theta_{k+1} (x_1, x_2, \dots, x_{k+1}) f_{k+1} (x_{k+1}) dx_{k+1}$ and $\int_{\mathbb{R}} \phi_{k+1} (x_1, x_2, \dots, x_{k+1}) f_{k+1} (x_{k+1}) dx_{k+1}$ are non-increasing functions, since $\theta_{k+1}(x_1, \dots, x_{k+1})$ and $\phi_{k+1}(x_1, \dots, x_{k+1})$ are non-increasing functions. Therefore, by the induction hypothesis,

$$E [E [\theta_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k] E [\phi_{k+1} (X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]]$$

$$\begin{aligned}
&= E \left[\left(\int_{\mathbb{R}} \theta_{k+1}(X_1, X_2, \dots, x_{k+1}) f_{k+1}(x_{k+1}) dx_{k+1} \right) \left(\int_{\mathbb{R}} \phi_{k+1}(X_1, X_2, \dots, x_{k+1}) f_{k+1}(x_{k+1}) dx_{k+1} \right) \right] \\
&\geq E \left[\left(\int_{\mathbb{R}} \theta_{k+1}(X_1, X_2, \dots, x_{k+1}) f_{k+1}(x_{k+1}) dx_{k+1} \right) \right] E \left[\left(\int_{\mathbb{R}} \phi_{k+1}(X_1, X_2, \dots, x_{k+1}) f_{k+1}(x_{k+1}) dx_{k+1} \right) \right] \\
&= E [E[\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] E [E[\phi_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]].
\end{aligned} \tag{10}$$

Step 4: We use the definition of conditional expectation to obtain the desired result.

$$\begin{aligned}
&E [E[\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] E [E[\phi_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] \\
&= E [\theta_{k+1}(X_1, X_2, \dots, X_{k+1})] E [\phi_{k+1}(X_1, X_2, \dots, X_{k+1})].
\end{aligned} \tag{11}$$

Combining equations (7), (9), (10) and (11), we obtain

$$\begin{aligned}
&E [\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) \phi_{k+1}(X_1, X_2, \dots, X_{k+1})] \\
&= E [E[\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) \phi_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] \\
&\geq E [E[\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k] E[\phi_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] \\
&\geq E [E[\theta_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] E [E[\phi_{k+1}(X_1, X_2, \dots, X_{k+1}) | X_1, X_2, \dots, X_k]] \\
&= E [\theta_{k+1}(X_1, X_2, \dots, X_{k+1})] E [\phi_{k+1}(X_1, X_2, \dots, X_{k+1})]
\end{aligned}$$

□.

In the following corollaries, we use Theorem 3 to develop a lower bound for the probability of an MLP. In 1, we prove the validity of this lower bound on essential networks where all aggregate edge costs are fixed costs. In Corollary 2, we generalize the result of Corollary 1 to essential networks with random aggregate edge costs.

COROLLARY 1. *Let E' be the set of aggregate edges in \tilde{G}_{st} , i.e., $\tilde{E}_{st} = \{e \in E_{st} : e \in r_{st}^*\} \cup E'$. Let $X_{i,i+1}$, $i = 1, \dots, |\tilde{V}|$, be independent real-valued random variables with probability density function $f_{i,i+1}$, respectively, representing the edge cost of r_{st}^* between nodes i and $i+1$ in \tilde{V}_{st} . Let $a_{ij} \in \mathbb{R}$, $(i,j) \in E'$, be the fixed edge cost of the aggregate path between nodes i and j in \tilde{V}_{st} . Then the probability of r_{st}^* is given by*

$$P \left(\bigcap_{(i,j) \in E'} \left\{ \sum_{k=i}^{j-1} X_{k,k+1} \leq a_{ij} \right\} \right)$$

and its corresponding lower bound by

$$\prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{k,k+1} \leq a_{ij} \right).$$

Proof of Corollary 1. Let $I_{ij}(x_{i,i+1}, \dots, x_{j-1,j})$, $(i,j) \in E'$ be indicator functions such that

$$I_{ij}(x_{i,i+1}, \dots, x_{j-1,j}) = \begin{cases} 1 & \text{if } \sum_{k=i}^{j-1} x_{k,k+1} \leq a_{ij} \\ 0 & \text{otherwise} \end{cases}.$$

Since $I_{ij}(x_{i,i+1}, \dots, x_{j-1,j})$ are non-increasing functions, we can apply Theorem 3 iteratively to obtain

$$\begin{aligned} P \left(\bigcap_{(i,j) \in E'} \left\{ \sum_{k=i}^{j-1} X_{k,k+1} \leq a_{ij} \right\} \right) &= E \left[\prod_{(i,j) \in E'} I_{ij}(X_{i,i+1}, \dots, X_{j-1,j}) \right] \\ &\geq \prod_{(i,j) \in E'} E [I_{ij}(X_{i,i+1}, \dots, X_{j-1,j})] \\ &= \prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{k,k+1} \leq a_{ij} \right). \end{aligned}$$

□.

In the following corollary, we generalize this probability lower bound to essential networks with random aggregate edge costs.

COROLLARY 2. *Let A_{ij} , $(i,j) \in E'$, be independent real-valued random variables with probability density function g_{ij} , respectively, representing the cost of the aggregate edge between nodes i and j in \tilde{G}_{st} . Then the probability of r_{st}^* is given by*

$$P \left(\bigcap_{(i,j) \in E'} \left\{ \sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij} \right\} \right) \quad (12)$$

and its corresponding lower bound by

$$\prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij} \right).$$

Proof of Corollary 2. The probability above (12) can be written as the following iterated integral:

$$\begin{aligned}
& P \left(\bigcap_{(i,j) \in E'} \left\{ \sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij} \right\} \right) \\
&= \int \dots \int P \left(\bigcap_{(i,j) \in E'} \left\{ \sum_{k=i}^{j-1} X_{k,k+1} \leq a_{ij} \right\} \right) \prod_{(i,j) \in E'} g_{ij}(a_{ij}) da_{ij} \\
&\geq \int \dots \int \prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{i,i+1} \leq a_{ij} \right) \prod_{(i,j) \in E'} g_{ij}(a_{ij}) da_{ij} \\
&= \int \dots \int \prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{i,i+1} \leq a_{ij} \right) g_{ij}(a_{ij}) da_{ij} \\
&= \prod_{(i,j) \in E'} \int P \left(\sum_{k=i}^{j-1} X_{i,i+1} \leq a_{ij} \right) g_{ij}(a_{ij}) da_{ij} \\
&= \prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{i,i+1} \leq A_{ij} \right), \tag{13}
\end{aligned}$$

where the inequality in (13) is obtained by applying the result of Corollary 2 \square .

It is important to note that the lower bound for the probability of r_{st}^* that we have developed in Corollary 2, *i.e.*,

$$P \left(\Phi_i = \min_{k \in K_{st}} \Phi_k \right) \geq \prod_{(i,j) \in E'} P \left(\sum_{k=i}^{j-1} X_{i,i+1} \leq A_{ij} \right),$$

can be computed efficiently by multiple one-dimensional integrals.

4.5. Upper Bound on Essential Series-Parallel networks

Now that we have obtained a lower bound for r_{st}^* that can be computed efficiently, we prove that a corresponding upper bound for (2) can also be computed by solving

$$\min_{k \in \tilde{K}_{st}} P(\Phi_{r_{st}^*} \leq \Phi_k). \tag{14}$$

The solution to problem (14) is a valid upper bound since for all $\tilde{k} \in \tilde{K}_{st} \setminus \{r_{st}^*\}$,

$$\bigcap_{k \in \tilde{K}_{st} \setminus \{r_{st}^*\}} \{ \Phi_{r_{st}^*} \leq \Phi_k \} \subset \{ \Phi_{r_{st}^*} \leq \Phi_{\tilde{k}} \}.$$

To solve (14), we replace all aggregate edge costs A_{ij} , $(i, j) \in E'$, with

$$p_{ij} = P \left(\sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij} \right),$$

and replace the edge cost of r_{st}^* between nodes i and $i+1$ with $P(X_{i,i+1} \leq X_{i,i+1}) = 1$. Problem (14) can then be restated as

$$\min_{k \in \tilde{K}_{st}} \prod_{(i,j) \in k} p_{ij}. \quad (15)$$

Since the logarithm is an increasing function on $(0, 1]$, we can restate (15) as

$$\exp \left[\min_{k \in \tilde{K}_{st}} \log \left(\prod_{(i,j) \in k} p_{ij} \right) \right] = \exp \left[\min_{k \in \tilde{K}_{st}} \sum_{(i,j) \in k} \log(p_{ij}) \right]. \quad (16)$$

Problem (16) now has the form of a deterministic shortest path problem, where the edge costs are

$$w_{ij} = \log \left(P \left(\sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij} \right) \right).$$

Since the probabilities are in $(0, 1]$, all edge costs are non-positive and bounded; but since essential series-parallel networks are acyclic, a solution to this deterministic shortest path problem is always well-defined. Consequently, we can solve this deterministic shortest path problem by standard methods and obtain our upper bound by exponentiating the resulting solution.

5. Computations

We generated series-parallel networks to test the SPMLP algorithm. We compared the SPMLP lower and upper bound results to the approximate probabilities obtained from Monte Carlo sampling, as well as their respective running times. We tested 10, 50, 100 and 250 edge networks with inputs of 25, 50 and 75 percent for (i) the average percentage of edges with fixed costs and (ii) the average percentage of series vs. parallel compositions. For each set of parameters, we tested four networks (144 total networks). These computations were performed on identical machines with Intel Pentium 4 CPUs with Hyper-threading, 3.0 GHz, 4096 MiB RAM.

Our experiment was designed to test the algorithm’s effectiveness at processing networks with significant noise and identifying paths of interest. Consequently, our bounds were exact in the majority of cases. Bound exactness should not be expected in general. Gaps arise in networks with alternate competitive paths, where taking one path eliminates the possibility of transferring to the other at a later node. In all our test cases, even when gaps were present, we successfully identified the MLP.

We designed Perl modules to generate random test networks. First, we build series-parallel network topologies via random combinations of series and parallel compositions. The inputs to the module are the number of edges and the ratio of series compositions to parallel ones. A second module then reads in the topology and randomly assigns either fixed or random costs to each edge of the network. The random costs are uniform random variables with integer lower bounds uniformly distributed between 0 and 9; and integer upper bounds uniformly distributed between the corresponding lower bound and 10. The fixed costs are integers uniformly distributed between 2 and 8, so that each fixed cost has an expected value that is both higher than the expected lower bound and lower than the expected upper bound of each random cost.

We implemented SPMLP using a combination of Perl and MATLAB. An MLP is identified via the dynamic sampling approach detailed in Section 4. The network topology is then simplified with respect to that solution, resulting in the corresponding essential series-parallel network. The essential network distributions are represented as discretized approximations to the actual continuous distributions, computed via Fast Fourier Transforms and point-wise products. For the discretized distributions, we use a uniform domain vector with grid spacing of 10^{-3} , in order to minimize both running time and computational error introduced by the discretization. The probabilities of interest are then computed by Riemann sum approximations from the discretized distributions. Finally, these probabilities are used to compute our lower and upper bounds.

In order to test the accuracy of the bounds produced, we implemented sequential Monte Carlo sampling with a desired sample standard deviation of 10^{-3} . If the desired sample standard deviation is not obtained within 10^6 iterations, the sampling is terminated. Out of the 144 networks tested, our

Mean Gap	Gap STD	Median Gap	Max Gap
0.006	0.032	0	0.230

Table 1 The optimality gaps are summarized in the table above. These gaps are computed as the differences between each probability upper bound computed by SPMLP and its corresponding lower bound.

Mean Gap MC-LB	Mean Gap UB-MC	Gap STD MC-LB	Gap STD UB-MC	Median Gap MC-LB	Median Gap UB-MC	Max Gap MC-LB	Max Gap UB-MC
0.004	0.002	0.020	0.012	0	0	0.150	0.093

Table 2 The optimality gaps are summarized in the table above. These gaps are computed as the differences between each probability bound computed by SPMLP and its corresponding Monte Carlo probability estimate.

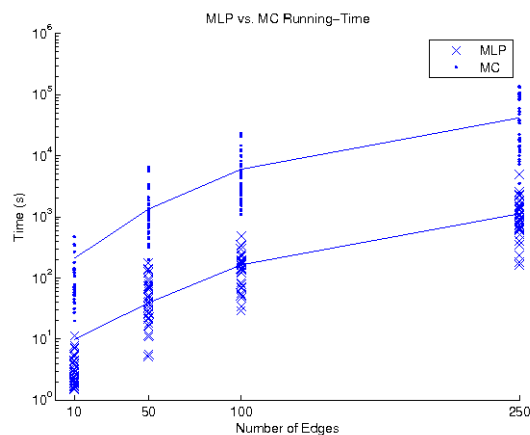


Figure 3 The average running time of Monte Carlo sampling is increasing at a greater rate than that of SPMLP, as is evidenced by the trend curves above. Moreover, the average running time of Monte Carlo sampling is approximately two orders of magnitude greater than that of SPMLP.

probability bounds yielded exact solutions in 134 cases. The ten cases where a gap is present result from conditions where there are nested aggregate paths in the essential network whose distributions are highly competitive with that of the projected MLP. The optimality gaps between the SPMLP bounds are summarized in Table 1. The gaps between the actual probabilities, computed via sequential Monte Carlo, and the bounds computed by SPMLP, are summarized in Table 2. The latter indicates that the gaps are not generally centered around the actual probability; rather the lower bounds are on average about twice as far from the actual probability than the corresponding upper bounds.

Figure 3 compares the running time of SPMLP versus that of sequential Monte Carlo sampling.

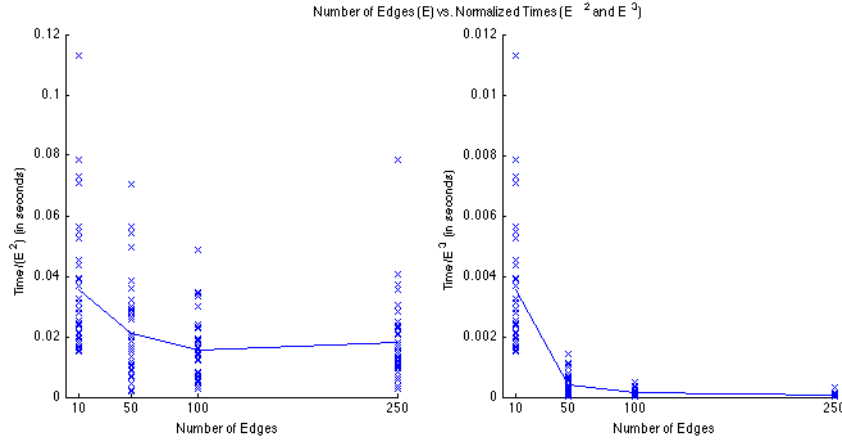


Figure 4 The curve on the left-hand plot shows that the actual running time for the MLPP is not $O(E^2)$, where E is the number of edges in the network. The downward-sloping curve on the right-hand plot shows that the running time for the MLPP grows at a slower rate than E^3 .

The average running time of the sequential Monte Carlo sampling is increasing at a greater rate than is the average running time of SPMLP, as is evidenced by the trend curves in Figure 3. Moreover, the average running time of Monte Carlo sampling is approximately two orders of magnitude greater than that of SPMLP. Notice that in practical terms, for a 250 node network, this is the difference between an average of about five minutes and an average of about 10 hours, a crucial advantage when the goal is to intercept an intruder before they can do harm.

The SPMLP has a polynomial running time based on the running times of Dijkstra’s algorithm, discretized Fast Fourier Transforms and numerical integration. In order to analyze the running time that was achieved in our computations, we plot normalized time as a function of network size. The two plots in Figure 4 show that the practical growth rate is between E^2 and E^3 , where E is the number of edges in the network.

6. Summary and Conclusions

For a given series-parallel network, with random edge costs, we have developed a dynamic sampling method for identifying an MLP. In doing so, we have utilized ordinal optimization in the context of stochastic network optimization. Additionally, we have established bounds on the objective for the MLPP. Through our computational results, we have verified that these bounds can be computed

efficiently. We have demonstrated that in many of the random networks tested, the lower and upper bounds agree, providing an exact solution.

A next step is to approximate more general networks which have series-parallel networks as subgraph isomorphisms. One method for accomplishing this may be to compute conditional distributions on the small portions of a network that prevent it from being series-parallel.

In future research, the methodological advances achieved here can be adapted for problems where one has cost distributions on individual locations or events and is interested in analyzing processes that result from alternative sequences of those events. Possible applications include homeland security, in which the MLPP can be used to detect routes most likely to be used by an intruder; and project management, in which the MLPP can be used to determine which alternative task subsequences are most likely to be responsible for project failures.

Acknowledgments

The authors would like to thank Robert Indik for his assistance in developing computational methods for continuous distributions. This material is based upon work supported by the National Science Foundation under Grant No. DMS-0602173.

References

- [1] V. G. Adlakha. An improved conditional Monte Carlo technique for the stochastic shortest path problem. *Management Science*, 32(10):1360–1367, 1986.
- [2] J. F. Bard and J. E. Bennett. Arc reduction and path preference in stochastic acyclic networks. *Management Science*, 37(2):198–215, 1991.
- [3] J. F. Bard and J. L. Miller. Probabilistic shortest path problems with budgetary constraints. *Computers and Operations Research*, 16(2):145–159, 1989.
- [4] H. Booth and R. E. Tarjan. Finding the minimum-cost maximum flow in a series-parallel network. *Journal of Algorithms*, 15(3):416–446, 1993.
- [5] J. M. Burt, Jr. and M. B. Garman. Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science*, 18(3):207–217, 1971.

-
- [6] J. T. Chayes, A. Puha, and T. Sweet. Independent and dependent percolation. *Probability Theory and Applications, IAS/Park City Mathematical Series*, 6:51–118, 1999.
- [7] D. W. Coit and A. E. Smith. Reliability optimization of series-parallel systems using a genetic algorithm. *Reliability, IEEE Transactions on*, 45(2):254–260, 1996.
- [8] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Inc. New York, NY, USA, 1998.
- [9] A. Eiger, P. B. Mirchandani, and H. Soroush. Path preferences and optimal paths in probabilistic networks. *Transportation Science*, 19(1):75–84, 1985.
- [10] P. C. Fishburn. Utility theory. *Management Science*, 14(5):335–378, 1968.
- [11] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971.
- [12] H. Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17(4):583–599, 1969.
- [13] Y. C. Ho, R. S. Sreenivas, and P. Vakili. Ordinal optimization of DEEDS. *Discrete Event Dynamic Systems*, 2(1):61–88, 1992.
- [14] A. Kasperski and P. Zieliński. The robust shortest path problem in series-parallel multidigraphs with interval data. *Operations Research Letters*, 34(1):69–76, 2006.
- [15] B. Klinz and G. J. Woeginger. Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43(3):153–162, 2004.
- [16] R. P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM*, 26(9):670–676, 1983.
- [17] R. Montemanni, L. M. Gambardella, and A. V. Donati. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32(3):225–232, 2004.
- [18] J. B. Sidney. The two-machine maximum flow time problem with series parallel precedence relations. *Operations Research*, 27(4):782–791, 1979.
- [19] C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg. The use of cutsets in Monte Carlo analysis of stochastic networks. *Mathematics and Computers in Simulation*, 21(4):376–384, 1979.
- [20] C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg. The stochastic shortest route problem. *Operations Research*, 28(5):1122–1129, 1980.

-
- [21] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 1–12, 1979.
- [22] J. A. Ward. Minimum-aggregate-concave-cost multicommodity flows in strong series-parallel networks. *Mathematics of Operations Research*, 24:106–129, 1999.
- [23] G. Yu and J. Yang. On the robust shortest path problem. *Computers and Operations Research*, 25(6): 457–468, 1998.
- [24] P. Zieliński. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158(3):570–576, 2004.