Stochastic Networks: Tractable Approaches for Identifying Strategic Paths

by

Daniel Reich

A Dissertation Submitted to the Faculty of the

Graduate Interdisciplinary Program
in Applied Mathematics

In Partial Fulfillment of the Requirements
For the Degree of

Doctor of Philosophy

In the Graduate College

The University of Arizona

2 0 0 9

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Daniel Reich

entitled Stochastic Networks: Tractable Approaches for Identifying Strategic Paths

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

_____ Date: Dec. 15, 2008
Leonardo Lopes

_____ Date: Dec. 15, 2008
Guzin Bayraksan

_____ Date: Dec. 15, 2008
Moysey Brio

_____ Date: Dec. 15, 2008
Robert Indik

_____ Date: Dec. 15, 2008
Simge Kucukyavuz

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: Dec. 15, 2008
Dissertation Director: Leonardo Lopes

Statement by Author

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Signed: Daniel Reich

Acknowledgments

I would like to begin by thanking my mom Joan and my dad Stephen; my brother Josh and my sister-in-law Linda; and my girlfriend Mia for all their love and support throughout the last five years.

I could not have asked for a better advisor than Leo Lopes. Leo has helped me improve professionally in almost every respect. He is always motivating and he has taught me that there are no setbacks in research, only opportunties for planning the next step forward. He has given me flexibility to shape the direction of our reseach, while always coming up with the needed suggestions to ensure that we stay on course. But Leo's mentoring goes well beyond this dissertation. When we started working together, Leo took an immediate interest in getting to know me personally and in understanding my priorities in life outside of work. So when he is constantly giving me advice, it is easy to see that he has more than just my career aspirations in mind. I cannot thank Leo enough for all he has done for me over the past few years. He is a great advisor and has become a close friend.

I am extremely fortunate to have four other outstanding committee members: Moysey Brio, Güzin Bayraksan, Robert Indik and Simge Küçükyavuz. They have all taught me a great deal; continually looked after my best interests; helped me find internships and jobs; provided guidance when I have needed it; and given me invaluable feedback on my research and this dissertation. I deeply appreciate all their help and support, and am truly grateful that I have been surrounded by so many terrific people.

I would like to mention several other people who have contributed to our research. Thanks to Linus Schrage for reviewing my dissertation; to Robert Maier for steering me towards Industrial Engineering; to Jan Wehr for identifying similarities between our work and percolation theory; to Bole Yang and and to Joari Costa for helping me obtain the best available data on Brazil's hydroelectric power system; to Tom Kennedy for teaching me probability theory; to Catherine Wang for sharing her expertise on clustering; to Aramian Wasielak for help on multiple fronts; and to the many other people who have helped me along the way.

Thank you to the Program in Applied Mathematics for providing me the opportunity and the funding to complete my degree. Thank you to the Department of Systems & Industrial Engineering for welcoming me into your community and treating me as one of your own graduate students. And thank you to the NSF. This material is based upon work supported by the National Science Foundation under Grant No. DMS-0602173.

I am lucky to have many great friends who have made my years as a graduate student such good ones and I would like to conclude my acknowledgments by thanking all of them.

Dedication

I would like to dedicate this work in memory of my grandparents Dorothy & Harold Reich and Gertrude & Wilbur Kropf. My grandparents all had a deep appreciation for education and I know they would have been very excited to see me complete my PhD.

Table of Contents

List of Figures 8

List of Tables 10

List of Algorithms 12

Abstract 13

1. The Most Likely Path Problem 14
    1.1. Introduction 14
    1.2. Formulation of the Most Likely Path Problem 17
        1.2.1. Redundant Fixed Costs and Degeneracy 19
        1.2.2. Subpath Optimality for the MLPP 20
    1.3. Series-Parallel Networks 22
    1.4. Algorithm for Solving the MLPP on Series-Parallel Networks . . 27
        1.4.1. SPMLP: Identifying the MLP on Series-Parallel Networks 27
        1.4.2. Dynamic Monte Carlo Sampling using Ordinal Optimization 27
        1.4.3. Essential Series-Parallel Networks 29
        1.4.4. Lower Bound on Essential Series-Parallel Networks . 30
        1.4.5. Upper Bound on Essential Series-Parallel networks . 36
    1.5. Computational Results 38
    1.6. Summary and Conclusions 42

2. Preprocessing Stochastic Shortest Path Problems on Directed Acyclic Networks 43
    2.1. Introduction 43
    2.2. Stochastic Shortest Path Problem Class 44
    2.3. SSPPC Preprocessing Algorithm 45
        2.3.1. Dominated Subpaths 48
        2.3.2. Subpath Constrained Shortest Path Problem 51
    2.4. Computational Results 56
    2.5. Summary and Conclusions 59

3. Empirical Effects of Clustering on Price Stability in a Stochastic Model for Power Grids. 62
    3.1. Introduction 62
    3.2. Clustering 64

TABLE OF CONTENTS—*Continued*

3.2.1. Background 64
3.2.2. Scenario Trees 65
3.2.3. Recursive Algorithm for Generating Scenario Trees via Clustering 65
3.3. Operation Planning 68
3.3.1. Stochastic Optimization Model 69
3.4. Computational Results 72
3.5. Summary and Conclusions 78

REFERENCES 81

List of Figures

1.1. This is an example of a degenerate network with two paths that have identical fixed costs of 5 and a third path that has a random cost uniformly distributed on [0,10]. 20
1.2. The network on the left is series-parallel. The one on the right is not, *i.e.* notice edge (1,3). 23
1.3. The network on the left is series-parallel. The one on the right is an essential series-parallel network and is formed from the one on the left. . 26
1.4. 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. . . 40
1.5. 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. 41

2.1. The SSPPC Preprocessing Algorithm illustrated for edge (2,3). (a) The original network with cost intervals on each edge. (b) $\overline{SP}_{1,5}$. (c) $\underline{\mathrm{SP}}_{1,5}|\overline{SP}_{1,5}$. (d) $\overline{SP}_{1,4}$. (e) $\underline{\mathrm{SP}}_{1,4}|\overline{SP}_{1,4}$. 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. 47
2.2. This shows that the converse of Lemma 2.1 is not true since $L_{qr}|\overline{SP}_{qr} = 19 < 20 = U_{qr}$. 49
2.3. 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 $1 \to 3 \to 4 \to 5 \to 7$ and has cost 26. In the LP relaxation flow is split between $1 \to 2 \to 3 \to 4 \to 5 \to 7$ and $1 \to 3 \to 4 \to 6 \to 7$ resulting in a lower cost of 25. 53
2.4. 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 $D = \{(1,4),(2,4)\}$. 56

List of Figures—*Continued*

2.5. The running time distribution of 50 tests networks for each edge size is shown above. The linear trend shows that the running time is increasingly at an approximately linear rate in the number of edges for a fixed number of nodes ($|V| = 48$). 59

3.1. The four plots above illustrate the relationship between dual prices and tree structure. 75

3.2. The histogram on the left provides an example of a tree whose distribution is highly symmetric: 1-16-1024. The histogram on the right provides an example of a tree whose distribution is not symmetric: 1-8-256-1024. . . 77

List of Tables

1.1. 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. 39
1.2. 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. . . 40

2.1. 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. 58
2.2. 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. 60

3.1. Summary of the dual prices for 60 sample trees for each scenario tree topology with 2 stages. Recall, 1-4 is a tree with a single node at time $t = 0$ and 4 nodes at $t = 1, \ldots, 11$. 73
3.2. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages and 256 leaves. Recall, 1-4-256 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 256 nodes at $t = 2, \ldots, 11$. 73
3.3. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages and 1024 leaves. Recall, 1-4-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$. 74
3.4. Summary of the dual prices for 60 sample trees for each scenario tree topology with 4 stages and 1024 leaves. Recall, 1-4-256-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$, 256 nodes at $t = 2$ and 1024 nodes at $t = 3, \ldots, 11$. 74
3.5. Summary of the dual prices for 60 sample trees for each scenario tree topology with 2 stages and no discount factor. Recall, 1-4 is a tree with a single node at time $t = 0$ and 4 nodes at $t = 1, \ldots, 11$. 78
3.6. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages, 256 leaves and no discount factor. Recall, 1-4-256 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 256 nodes at $t = 2, \ldots, 11$. 78

List of Tables—*Continued*

3.7. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages, 1024 leaves and no discount factor. Recall, 1-4-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$. 79
3.8. Summary of the dual prices for 60 sample trees for each scenario tree topology with 4 stages, 1024 leaves and no discount factor. Recall, 1-4-256-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$, 256 nodes at $t = 2$ and 1024 nodes at $t = 3, \ldots, 11$. 79

List of Algorithms

1.1. Reduce any series-parallel network to an essential series-parallel network. 24

2.1. The SSPPC Preprocessing Algorithm takes a directed acyclic network G_{st} and preprocess out any non-weak edges that are detected. 55

3.1. Generate a scenario tree from arrays S and Q of m independent time-series scenarios, where S contains the original time-series, Q contains the corresponding normalized and reweighted time-series, $t = 1$ and $v = \emptyset$ initially, and $\vec{k}$ provides a lower bound for the number of nodes to be generated at each stage of the tree. 67

Abstract

In Chapter 1, 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.

In Chapter 2, 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 in the test problems remain unclassified after preprocessing.

In Chapter 3, we introduce a methodology for generating scenario trees from independent samples via k-means clustering. The trees are then input into a well-known stochastic optimization model used for operation planning on the Brazilian power system. The dual prices from this model are by contract used in Brazil to determine prices paid to operators of hydroelectric energy plants. Our computational results provide key insights regarding the variability introduced into these dual prices by both the choice of scenario tree topology and by k-means clustering. We show that for sufficiently large scenario trees, the dual prices are quite stable.

1. The Most Likely Path Problem

1.1. Introduction

In this chapter, 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 (Frank 1969, Sigal et al. 1980). 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 (Yu and Yang 1998, Zieliński 2004, Montemanni et al. 2004). 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 (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 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 (1983) uses this technique to extend the von Neumann-Morgenstern (see Fishburn 1968) formulation for identifying ideal methods of evaluation in the presence of uncertainty. Bard and Miller (1989) 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 (1991) and Eiger et al. (1985).

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 (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. Consequently, he provides a sampling based method for statistically analyzing the distribution functions. Burt and Garman (1971) develop an alternate sampling method that uses conditional Monte Carlo simulation for estimating distribution functions along paths in stochastic networks. Adlakha (1986) combines ideas from the cut-set approach of Sigal et al. (1979) with aspects of the simulation method of Burt and Garman (1971) to compute distribution functions of the minimum cost paths in stochastic networks.

Sigal et al. (1980) 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 (1969) aims to identify the distribution of the minimum cost path, the formulation of Sigal et al. (1980) 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. (1980) 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 exponential 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. (1980) 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. (1979) 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 (Coit and Smith 1996, Klinz and Woeginger 2004, Sidney 1979, Ward 1999, Booth and Tarjan 1993).

The remainder of this chapter is organized as follows. In Section 1.2, we provide

background graph-theoretic concepts, formulate the MLPP and prove subpath optimality for the MLPP on general networks. Section 1.3 provides a formal definition of series-parallel networks and introduces our new subclass of *essential series-parallel networks*. Section 1.4 presents our computationally tractable algorithm for identifying the MLP and bounding its probability on series-parallel networks. Section 1.5 summarizes our computational results. Finally, Section 1.6 presents consequences of our work and future research directions.

1.2. 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 (see Cook et al. (1998)) 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 specified 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 chapter we use the notation in Cook et al. (1998) 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 1.1 (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 chapter, 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 necessarilly unique) with highest probability of being the shortest path, *i.e.*, a path r^*_{st} satisfying

$$r^*_{st} \in \arg\max_{i \in K_{st}} P\left(\Phi_i = \min_{k \in K_{st}} \Phi_k\right), \tag{1.1}$$

where

$$P\left(\Phi_i = \min_{k \in K_{st}} \Phi_k\right) = P\left(\bigcap_{k \in K_{st}} \{\Phi_i \leq \Phi_k\}\right). \tag{1.2}$$

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 (1.1), which in turn requires multivariate integration. Moreover, since for many common topologies the number of paths in a given network may scale exponentially with its node or edge size, solving for the objective in (1.1) by enumerating all paths is in general 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.

1.2.1. 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

$$\sum_{i \in K_{st}} P\left(\Phi_i = \min_{k \in K_{st}} \Phi_k\right) > 1 \tag{1.3}$$

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 (1.3) holds in degenerate networks; and why these networks should be avoided from a modeling perspective.

Consider the degenerate network in Figure 1.1 with three independent paths (no shared arcs between 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

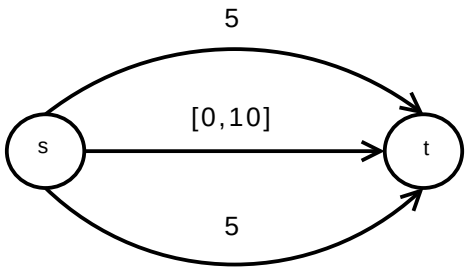

FIGURE 1.1. This is an example of a degenerate network with two paths that have identical fixed costs of 5 and a third path that has a random cost uniformly distributed on [0,10].

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.

1.2.2. 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 1.1. *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. We show that if there exists a path $\tilde{r}_{uv}$ such that

$$P\left(\Phi_{\tilde{r}_{uv}} = \min_{k \in K_{uv}} \Phi_k\right) > P\left(\Phi_{r^*_{uv}} = \min_{k \in K_{uv}} \Phi_k\right) \tag{1.4}$$

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

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

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

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

Consequently, if equation (1.4) holds then $\tilde{r}_{st}$ is more likely than r^*_{st}, which is a contradiction. Thus r^*_{uv} is optimal for G_{uv}.

□

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 (1.1) cannot be computed iteratively, due to event dependence. We will see that in series-parallel networks, this event dependence can be handled iteratively.

1.3. 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 1.2 (Series-Parallel Networks (Booth and Tarjan 1993))**.** 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:

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 $G_{s_1t_1}(V_{s_1t_1}, E_{s_1t_1})$ and $G_{s_2t_2}(V_{s_2t_2}, E_{s_2t_2})$ 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 $G_{s_pt_p}$.

3. *Series Composition*: By setting $t_1 = s_2$ or $t_2 = s_1$, we form the series-parallel network $G_{s_1t_2}$or $G_{s_2t_1}$, respectively.

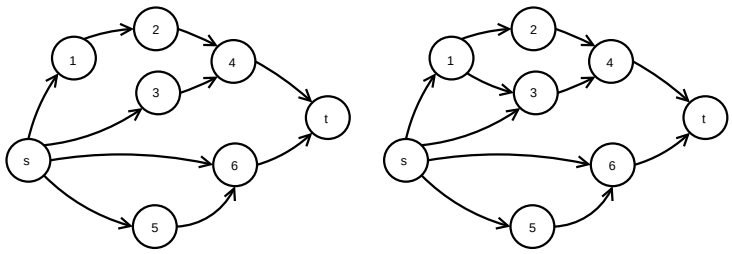

FIGURE 1.2. The network on the left is series-parallel. The one on the right is not, *i.e.* notice edge (1,3).

Figure 1.2 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 1.3 (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 $f : \tilde{V}_{st} \to \mathbb{N}$, such that $f(s) = 1$, $f(t) = |\tilde{V}_{st}|$, $f(s) < f(v) < f(t)$ for all $v \in \tilde{V}_{st} \backslash \{s, t\}$, 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 1.3 provides an example of an essential series-parallel network.

Algorithm 1.1 Reduce any series-parallel network to an essential series-parallel network.

Choose any $s-t$ path $k \in K_{st}$.
$\tilde{V}_{st} = V_{st}$.
$\tilde{E}_{st} = E_{st}$.
Flag = 0.
while Flag == 0 **do**
  Flag = 1.
  for all $(u,v) \in \tilde{E}_{st} \backslash k$ **do**
    if there exists more than one edge $(u,v) \in \tilde{E}_{st} \backslash k$ **then**
      Parallel Decomposition: Remove from $\tilde{E}_{st}$ all but one edge $(u,v) \in \tilde{E}_{st} \backslash k$.
      Flag = 0.
    end if
  end for
  for all $(u,v) \in \tilde{E}_{st} \backslash k$ **do**
    if $\exists q \in \tilde{V}_{st}$ such that $(q,u) \in \tilde{E}_{st} \backslash k$ **then**
      if $\nexists q' \in \tilde{V}_{st}$ such that $(q',u) \in \tilde{E}_{st} \backslash (q,u)$ and $\nexists v' \in \tilde{V}_{st}$ such that $(u,v') \in \tilde{E}_{st} \backslash (u,v)$ **then**
        Series Decomposition: $\tilde{E}_{st} = \{\tilde{E}_{st} \backslash \{(q,u),(u,v)\}\} \cup \{(q,v)\}$. $\tilde{V}_{st} = \tilde{V}_{st} \backslash \{u\}$.
        Flag = 0.
      end if
    else if $\exists r \in \tilde{V}_{st}$ such that $(v,r) \in \tilde{E}_{st} \backslash k$ **then**
      if $\nexists u' \in \tilde{V}_{st}$ such that $(u',v) \in \tilde{E}_{st} \backslash (u,v)$ and $\nexists r' \in \tilde{V}_{st}$ such that $(v,r') \in \tilde{E}_{st} \backslash (v,r)$ **then**
        Series Decomposition: $\tilde{E}_{st} = \{\tilde{E}_{st} \backslash \{(u,v),(v,r)\}\} \cup \{(u,r)\}$. $\tilde{V}_{st} = \tilde{V}_{st} \backslash \{v\}$.
        Flag = 0.
      end if
    end if
  end for
end while
return $\tilde{G}_{st}$

Essential series-parallel networks can be obtained by deconstructing parts of a series-parallel network through combinations of *parallel and series decompositions*. These decompositions are a reversal of the rules provided in Definition 1.2. This process is formalized in Algorithm 1.1.

Theorem 1.2. *Any series-parallel network G_{st} can be reduced to an essential series-parallel network $\tilde{G}_{st}$ via Algorithm 1.1.*

Proof. 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 $k \in K_{st}$. 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 $(u', v) \in \tilde{E}_{st} \backslash k$ for $u' \in \tilde{V}_{st}$; and there exists at least one edge $(v, r') \in \tilde{E}_{st} \backslash k$ for $r' \in \tilde{V}_{st}$.

Let $E^{\text{in}}_{.v}$ be the set of edges $\{(u', v) \in \tilde{E}_{st} \backslash k : u' \in \tilde{V}_{st}\}$. Since G_{st} is series-parallel, v must be the sink of some series-parallel network $G_{.v}$ whose edge set contains $E^{\text{in}}_{.v}$. Since the edges in k are consecutively connected, the sink of $G_{.v}$ would have to be connected to one of the edges in k if $G_{.v}$ contained any edge in k. Therefore, by (a) above, $G_{.v}$ must not contain any edge in k. Consequently, by Definition 1.2 (series-parallel networks), $G_{.v}$ 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 $\{(v, r') \in \tilde{E}_{st} \backslash k : r' \in \tilde{V}_{st}\}$. Hence, without loss of generality we can consider the case (b) above when there is exactly one edge $(u', v) \in \tilde{E}_{st} \backslash k$ for some $u' \in \tilde{V}_{st}$ and one edge $(v, r') \in \tilde{E}_{st} \backslash k$ for some $r' \in \tilde{V}_{st}$. However, these two edges are replaced by a single edge during the series decompostion step and node v is removed, which is a contradiction to our assumption.

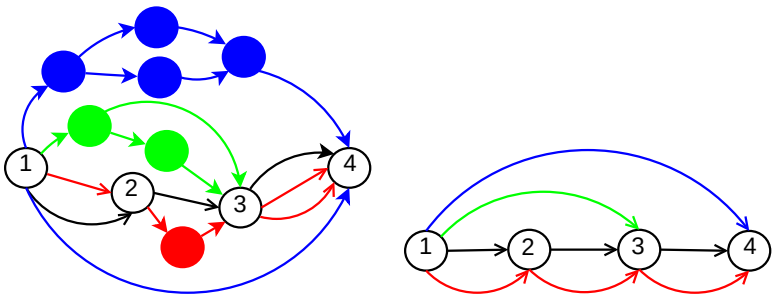

FIGURE 1.3. The network on the left is series-parallel. The one on the right is an essential series-parallel network and is formed from the one on the left.

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 1.3 (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.

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 1.1.

□

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

1.4.1. 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 interative process that can be used to evaluate the joint probability measure in our objective (1.1). 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 (1.2) 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 (1.1), 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.

1.4.2. Dynamic Monte Carlo Sampling using Ordinal Optimization

Consider two subpaths r^1_{uv} and r^2_{uv} 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^1_{uv} or r^2_{uv} as a potential subpath of the optimal path r^*_{st}, we need not precisely measure both sides of inequality (1.4); rather we need only identify which side is greater.

Ho et al. (1992) 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 (1.4) 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 (1.4) 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 1.1, 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 a high degree of accuracy in $\mathrm{O}(|E|)$ 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 (1.2) 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 probabil-

ity. Before describing the method for obtaining the essential series-parallel network of interest, let us introduce the necessary probability background information.

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

$$F_C(x) = 1 - ([1 - F_X(x)][1 - F_Y(x)]), \tag{1.5}$$

where F_C is the cumulative distribution of C.

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

$$P(X+Y \leq c) = \int\int_{x+y\leq c} f_X(x)f_Y(y)dxdy = \int_{-\infty}^{\infty} F_X(c-y)f_Y(y)dy = F_X \star f_Y. \tag{1.6}$$

1.4.3. 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 1.1.

Through Algorithm 1.1, 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 (1.6). At each parallel decomposition, the cost distribution of the

remaining edge is found by applying the multiplication operator (1.5). By construction, the probability measure of interest is unchanged under both series and parallel decompositions.

1.4.4. Lower Bound on Essential Series-Parallel Networks

We say that two events A and B are *positively correlated* if

$$P(A \cap B) \geq P(A)P(B). \tag{1.7}$$

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 (1.2) on the essential network into multiple one-dimensional integrals, thereby producing a valid lower bound.

In order to show the needed inequality (1.7), *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 1.3 is analogous to the proof by Chayes et al. (1999) of the Harris-FKG Inequality (Fortuin et al. 1971), which is commonly used in percolation theory.

Theorem 1.3. *Let* $\theta_n : \mathbb{R}^n \to \mathbb{R}$ *and* $\phi_n : \mathbb{R}^n \to \mathbb{R}$ *be* non-increasing *functions, i.e. for every* i*,*

$$\theta_n(x_1, \ldots, x_i, \ldots, x_n) \leq \theta_n(x_1, \ldots, \tilde{x}_i, \ldots, x_n)$$

if $x_i \geq \tilde{x}_i$ *and*

$$\phi_n(x_1, \ldots, x_i, \ldots, x_n) \leq \phi_n(x_1, \ldots, \tilde{x}_i, \ldots, x_n)$$

if $x_i \geq \tilde{x}_i$*. Let* $X_1, \ldots, X_n$ *be independent real-valued random variables with probability density functions* $f_1, f_{2,\ldots,} f_n$*, respectively. Then*

$$\begin{aligned} &E\left[\theta_n\left(X_1, \ldots, X_n\right) \phi_n\left(X_1, \ldots, X_n\right)\right] \\ &\geq E\left[\theta_n\left(X_1, \ldots, X_n\right)\right] E\left[\phi_n\left(X_1, \ldots, X_n\right)\right]. \end{aligned}$$

.

Proof.

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:

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

So,

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

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 $X_1, \ldots, X_k$ 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 $X_1, \ldots, X_k$ to obtain an outer expectation over a random variable of dimension k and an inner expectation over a random variable of dimension 1.

$$\begin{aligned} &E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) \phi_{k+1}\left(X_1, \ldots, X_{k+1}\right)\right] \\ &= E\left[E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) \phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right]\right], \end{aligned} \tag{1.8}$$

where the inner expectation,

$$\begin{aligned} &E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) \phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right] \\ &= \int_{\mathbb{R}} \theta_{k+1}\left(X_1, \ldots, X_k, x_{k+1}\right) \phi_{k+1}\left(X_1, \ldots, X_k, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}. \end{aligned} \tag{1.9}$$

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

$$\begin{aligned} & E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) \phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right] \\ & \geq E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right] E\left[\phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right]. \end{aligned} \tag{1.10}$$

Notice that the resulting product of one-dimensional expectations (1.10) 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

$$\int_{\mathbb{R}} \theta_{k+1}\left(x_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}$$

and

$$\int_{\mathbb{R}} \phi_{k+1}\left(x_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}$$

are non-increasing functions, since $\theta_{k+1}(x_1, \ldots, x_{k+1})$ and $\phi_{k+1}(x_1, \ldots, x_{k+1})$ are non-increasing functions. Therefore, by the induction hypothesis,

$$\begin{aligned} & E\left[E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right] E\left[\phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right]\right] \\ & = E\Bigg[\left(\int_{\mathbb{R}} \theta_{k+1}\left(X_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}\right) \times \\ & \qquad \left(\int_{\mathbb{R}} \phi_{k+1}\left(X_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}\right)\Bigg] \qquad (1.11) \\ & \geq E\left[\left(\int_{\mathbb{R}} \theta_{k+1}\left(X_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}\right)\right] \times \\ & \qquad E\left[\left(\int_{\mathbb{R}} \phi_{k+1}\left(X_1, \ldots, x_{k+1}\right) f_{k+1}\left(x_{k+1}\right) dx_{k+1}\right)\right] \qquad (1.12) \\ & = E\left[E\left[\theta_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right]\right] E\left[E\left[\phi_{k+1}\left(X_1, \ldots, X_{k+1}\right) | X_1, \ldots, X_k\right]\right]. \qquad (1.13) \end{aligned}$$

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

$$E\left[E\left[\theta_{k+1}\left(X_1,\ldots,X_{k+1}\right)|X_1,\ldots,X_k\right]\right]E\left[E\left[\phi_{k+1}\left(X_1,\ldots,X_{k+1}\right)|X_1,\ldots,X_k\right]\right]$$
$$=E\left[\theta_{k+1}\left(X_1,\ldots,X_{k+1}\right)\right]E\left[\phi_{k+1}\left(X_1,\ldots,X_{k+1}\right)\right]. \tag{1.14}$$

Combining equations (1.8), (1.10), (1.13) and (1.14), we obtain

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

□

In the following corollaries, we use Theorem 1.3 to develop a lower bound for the probability of an MLP. In Corollary 1.1, we prove the validity of this lower bound on essential networks where all aggregate edge costs are fixed costs. In Corollary 1.2, we generalize the result of Corollary 1.1 to essential networks with random aggregate edge costs.

Corollary 1.1. *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 $X_{i,i+1}$, $i=1,\ldots,|\tilde{V}|$, 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

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

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

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

Since $I_{ij}(x_{i,i+1},\ldots,x_{j-1,j})$ are non-increasing functions, we can apply Theorem 1.3 iteratively to obtain

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

□

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

Corollary 1.2. *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*

$$P\left(\bigcap_{(i,j)\in E'}\left\{\sum_{k=i}^{j-1} X_{k,k+1} \leq A_{ij}\right\}\right) \tag{1.15}$$

and its corresponding lower bound by

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

Proof. The probability above (1.15) can be written as the following iterated integral:

$$
\begin{aligned}
& P\left(\bigcap_{(i,j)\in E'}\left\{\sum_{k=i}^{j-1} X_{k,k+1}\le A_{ij}\right\}\right)\\
&=\int\cdots\int P\left(\bigcap_{(i,j)\in E'}\left\{\sum_{k=i}^{j-1} X_{k,k+1}\le a_{ij}\right\}\right)\prod_{(i,j)\in E'} g_{ij}\left(a_{ij}\right)da_{ij}\\
&\ge\int\cdots\int\prod_{(i,j)\in E'} P\left(\sum_{k=i}^{j-1} X_{i,i+1}\le a_{ij}\right)\prod_{(i,j)\in E'} g_{ij}\left(a_{ij}\right)da_{ij}\\
&=\int\cdots\int\prod_{(i,j)\in E'} P\left(\sum_{k=i}^{j-1} X_{i,i+1}\le a_{ij}\right) g_{ij}\left(a_{ij}\right)da_{ij}\\
&=\prod_{(i,j)\in E'}\int P\left(\sum_{k=i}^{j-1} X_{i,i+1}\le a_{ij}\right) g_{ij}\left(a_{ij}\right)da_{ij}\\
&=\prod_{(i,j)\in E'} P\left(\sum_{k=i}^{j-1} X_{i,i+1}\le A_{ij}\right), \qquad (1.16)
\end{aligned}
$$

where the inequality in (1.16) is obtained by applying the result of Corollary 1.2.

□

It is important to note that the lower bound for the probability of r^*_{st} that we have developed in Corollary 1.2, *i.e.*,

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

can be computed efficiently by multiple one-dimensional integrals.

1.4.5. Upper Bound on Essential Series-Parallel networks

Now that we have obtained a lower bound for $P(r^*_{st}=\min_{k\in K_{st}}\Phi_k)$ that can be computed efficiently, we prove that a corresponding upper bound for (1.2) can also be computed by solving

$$\min_{k \in \tilde{K}_{st}} \; P\left(\Phi_{r^*_{st}} \leq \Phi_k\right). \tag{1.17}$$

The solution to problem (1.17) is a valid upper bound since for all $\tilde{k} \in \tilde{K}_{st} \backslash \{r^*_{st}\}$,

$$\bigcap_{k \in \tilde{K}_{st}} \left\{\Phi_{r^*_{st}} \leq \Phi_k\right\} \subseteq \left\{\Phi_{r^*_{st}} \leq \Phi_{\tilde{k}}\right\}.$$

To solve (1.17), 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 (1.17) can then be restated as

$$\min_{k \in \tilde{K}_{st}} \prod_{(i,j) \in k} p_{ij}. \tag{1.18}$$

Since the logarithm is an increasing function on $(0,1]$, we can restate (1.18) as

$$\exp\left[\min_{k \in \tilde{K}_{st}} \; \log\left(\prod_{(i,j) \in k} p_{ij}\right)\right] = \exp\left[\min_{k \in \tilde{K}_{st}} \; \sum_{(i,j) \in k} \log\left(p_{ij}\right)\right]. \tag{1.19}$$

Problem (1.19) 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.

1.5. Computational Results

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 succesfully 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

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

TABLE 1.1. 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.

identified via the dynamic sampling approach detailed in Section 1.4. 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 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 1.1. The gaps between the actual probabilities, computed via sequential Monte Carlo, and the bounds computed by SPMLP, are summarized in Table 1.2. 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.

Mean Gap MC-LB	Gap STD MC-LB	Median Gap MC-LB	Max Gap MC-LB
0.004	0.020	0	0.150

Mean Gap MC-UB	Gap STD MC-UB	Median Gap MC-UB	Max Gap MC-UB
0.002	0.012	0	0.093

TABLE 1.2. 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.

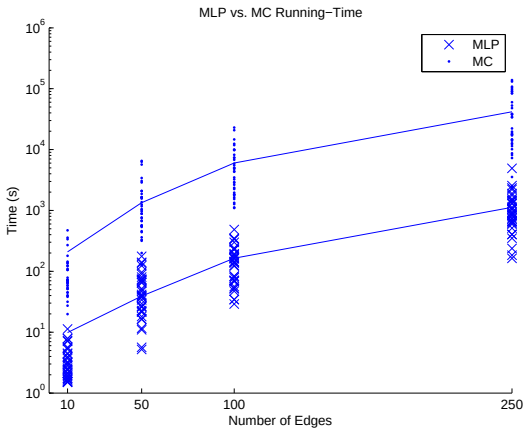

FIGURE 1.4. The average running time of Monte Carlo sampling is increasing at a greater rate than that of SPMLP, as is evidenced by the trend curves above. Moreover, the average running time of Monte Carlo sampling is approximately two orders of magnitude greater than that of SPMLP.

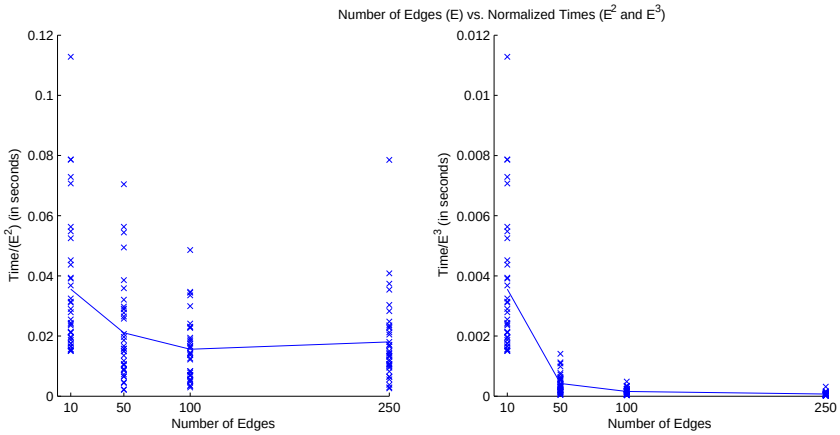

FIGURE 1.5. The curve on the left-hand plot shows that the actual running time for the MLPP is not $O(E^2)$, where E is the number of edges in the network. The downward-sloping curve on the right-hand plot shows that the running time for the MLPP grows at a slower rate than E^3.

Figure 1.4 compares the running time of SPMLP versus that of sequential Monte Carlo sampling. 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 1.4. 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.

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 1.5 show that the practical growth rate is between E^2 and E^3, where E is the number of edges in the network.

1.6. 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.

2. Preprocessing Stochastic Shortest Path Problems on Directed Acyclic Networks

2.1. 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 chapter 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 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. In the introduction to Chapter 1, we provide references for many of these other frameworks.

Preprocessing edges significantly increases the utility of many existing NP-hard stochastic shortest path frameworks by shrinking problem sizes. 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 et al. (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 chapter is organized as follows. In Section 2.2, we provide background graph-theoretic concepts and define the class of stochastic shortest path problems for which our preprocessing method applies. Section 2.3 presents our deterministic algorithm for preprocessing directed acyclic networks. Section 2.4 summarizes our computational results. Finally, Section 2.5 presents consequences of our work and future research directions.

2.2. 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 $[\underline{c}_e, \overline{c}_e]$ 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 $c_e^\omega \in [\underline{c}_e, \overline{c}_e]$, $e \in E$; and the cost of any given set of edges E under scenario ω is given by

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

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

Definition 2.1 (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 $[\underline{c}_e, \overline{c}_e]$. 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.

Solving SSPPC problems is difficult on general networks, since for many common topologies the number of paths in a given network may scale exponentially with its node or edge size. Our preprocessing method aims to reduce the needed computational time on general directed acyclic networks by eliminating unnecessary parts of the network deterministically.

2.3. SSPPC Preprocessing Algorithm

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

Definition 2.2 (Weak and Non-weak Edges)**.** Let SP^{ω}_{qr} 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^{ω}_{qr}. 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 2.3 and 2.4 provide a foundation for isolating e against its competition in various subnetworks G_{qr} of G_{st}.

Definition 2.3 ($q-r$ 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 $\overline{c}$ 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 2.4 ($q-r$ Subnetwork Lower Bound Path)**.** A $q-r$ *subnetwork lower bound path* $\underline{\text{SP}}_{qr}|\overline{SP}_{qr}$ with cost $L_{qr}|\overline{SP}_{qr}$ is formed by joining at edge $e = (u, v)$ 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 $E_{qr} \backslash \overline{SP}_{qr}$ have lower bound costs. In other words, $\underline{\text{SP}}_{qr}|\overline{SP}_{qr}$ 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 2.1. Steps (b) - (f) leverage the subnetwork upper bound and lower bound paths defined above. The method for step (g) will be introduced in Subsection 2.3.2.

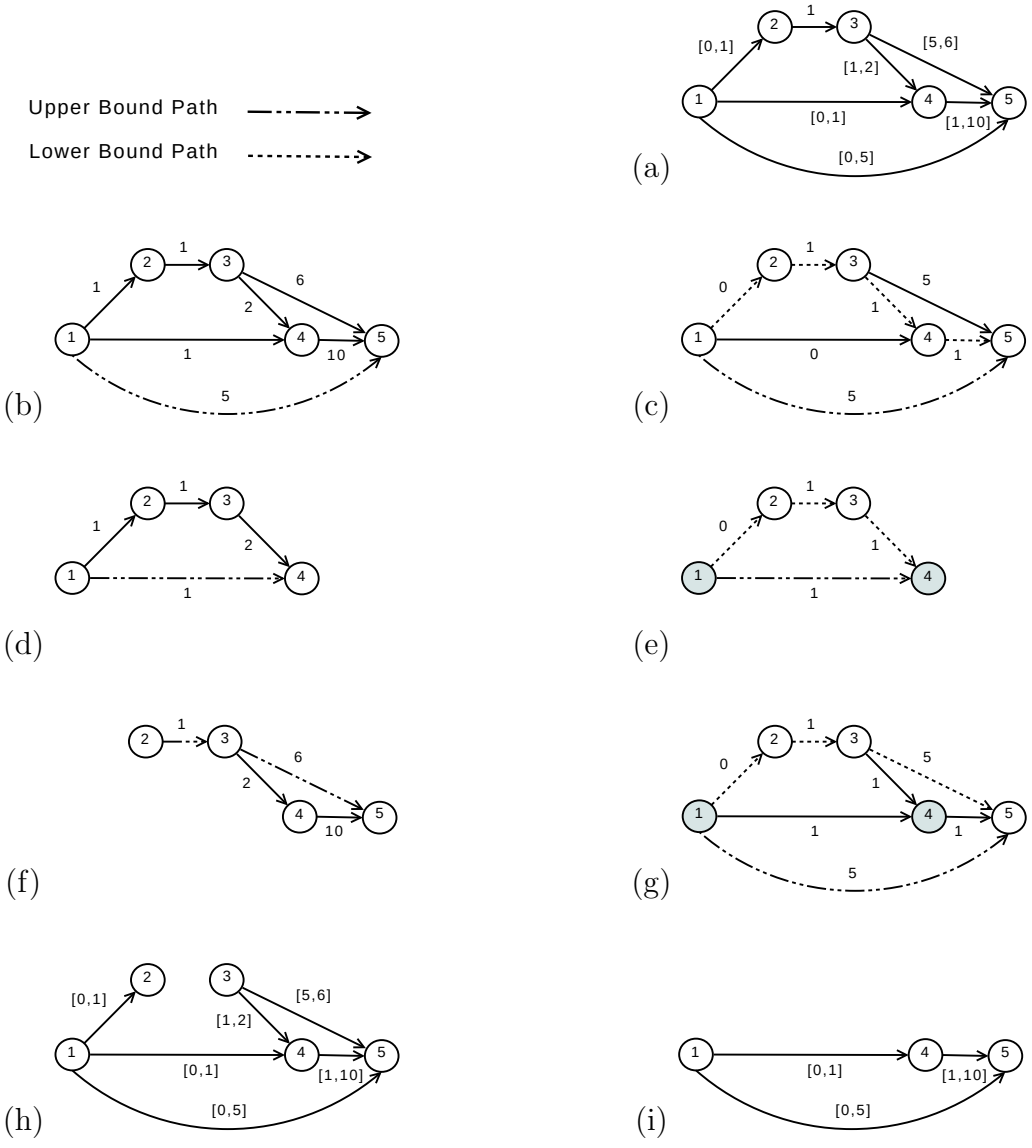

FIGURE 2.1. The SSPPC Preprocessing Algorithm illustrated for edge (2,3). (a) The original network with cost intervals on each edge. (b) $\overline{SP}_{1,5}$. (c) $\underline{\mathrm{SP}}_{1,5}|\overline{SP}_{1,5}$. (d) $\overline{SP}_{1,4}$. (e) $\underline{\mathrm{SP}}_{1,4}|\overline{SP}_{1,4}$. 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.

2.3.1. Dominated Subpaths

We justify step (e) in Figure 2.1 by showing that if a subnetwork lower bound path $\underline{\mathrm{SP}}_{qr}|\overline{SP}_{qr}$ has higher cost than its corresponding upper bound path $\overline{SP}_{qr}$, then e is non-weak on G_{qr}.

Lemma 2.1. *If $L_{qr}|\overline{SP}_{qr} > U_{qr}$ then $c^{\omega}(SP^{\omega}_{qu}) + \underline{c}_e + c^{\omega}(SP^{\omega}_{vr}) > c^{\omega}(SP^{\omega}_{qr})$ for any $\omega \in \Omega$.*

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

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

Consequently,

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

□

Note: The converse of Lemma 2.1 is not true, as can be seen in Figure 2.2, where

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

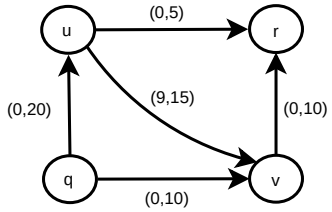

FIGURE 2.2. This shows that the converse of Lemma 2.1 is not true since $L_{qr}|\overline{SP}_{qr} = 19 < 20 = U_{qr}$.

but for all ω,

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

since

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

We justify step (g) in Figure 2.1 by Corollary 2.1 below. If a subnetwork lower bound path $\underline{\text{SP}}_{qr}|\overline{SP}_{qr}$ has higher cost than its corresponding upper bound path $\overline{SP}_{qr}$, 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.

Corollary 2.1. *If $L_{qr}|\overline{SP}_{qr} > U_{qr}$, then for every ω, $c^{\omega}(SP^{\omega}_{sq}) + c^{\omega}(SP^{\omega}_{qu}) + \underline{c}_e + c^{\omega}(SP^{\omega}_{vr}) + c^{\omega}(SP^{\omega}_{rt}) > c^{\omega}(SP^{\omega}_{st})$.*

Proof. By Lemma 2.1, if $L_{qr}|\overline{SP}_{qr} > U_{qr}$ then $c^{\omega}(SP^{\omega}_{qu}) + \underline{c}_e + c^{\omega}(SP^{\omega}_{vr}) > c^{\omega}(SP^{\omega}_{qr})$. Consequently, for all ω,

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

□

Lemma 2.1 and Corollary 2.1 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 2.5 and Lemma 2.2 formalize this idea.

Definition 2.5 ($q-r$ Global Lower Bound Path)**.** A $q-r$ *global lower bound path* $\underline{\mathrm{SP}}_{sqrt}|\overline{SP}_{st}$ with cost $L_{sqrt}|\overline{SP}_{st}$ 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 $E_{st}\backslash\overline{SP}_{st}$ have lower bound costs. In other words, $\underline{\mathrm{SP}}_{sqrt}|\overline{SP}_{st}$ 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 2.2 below that if a global lower bound path $\underline{\mathrm{SP}}_{sqrt}|\overline{SP}_{st}$ 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 2.2. *If* $L_{sqrt}|\overline{SP}_{st} > U_{st}$*, then for every* ω*,* $c^{\omega}(SP^{\omega}_{sq}) + c^{\omega}(SP^{\omega}_{qu}) + \underline{c}_e + c^{\omega}(SP^{\omega}_{vr}) + c^{\omega}(SP^{\omega}_{rt}) > c^{\omega}(SP^{\omega}_{st})$.

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

$$
\begin{aligned}
& c^{\omega}(SP^{\omega}_{sq}) + c^{\omega}(SP^{\omega}_{qu}) + \underline{\mathrm{c}}_e + c^{\omega}(SP^{\omega}_{vr}) + c^{\omega}(SP^{\omega}_{rt}) + \overline{c}(E^{\omega}_{st}) - c^{\omega}(E^{\omega}_{st}) \\
=& c^{\omega}(SP^{\omega}_{sq} \cup SP^{\omega}_{qu} \cup \{e\} \cup SP^{\omega}_{vr} \cup SP^{\omega}_{rt}\backslash E^{\omega}_{st}) + \overline{c}(E^{\omega}_{st}) \\
\geq& \underline{\mathrm{c}}(SP^{\omega}_{sq} \cup SP^{\omega}_{qu} \cup \{e\} \cup SP^{\omega}_{vr} \cup SP^{\omega}_{rt}\backslash E^{\omega}_{st}) + \overline{c}(E^{\omega}_{st}) \\
\geq& L_{sqrt}|\overline{SP}_{st}.
\end{aligned}
$$

Consequently,

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

□

Lemma 2.1, Corollary 2.1 and Lemma 2.2 provide conditions for restricting certain subpaths containing e in the given network G_{st}. The following definition characterizes these restrictions.

Definition 2.6 (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 $L_{sqrt}|\overline{SP}_{st} > U_{st}$. 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}.

2.3.2. 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 $(q, r) \in D$.

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 $E_{st} \backslash \overline{SP}_{st}$ 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^*.

$$\text{Minimize} \quad \sum_{(i,j)\in E} c_{ij} x_{ij} \tag{2.1}$$

subject to

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{for all } i \notin \{s, t\} \\ -1 & \text{if } i = t \end{cases} \tag{2.2}$$

$$\sum_{j:(q,j)\in E} x_{qj} + \sum_{j:(j,r)\in E} x_{jr} \leq 1 \quad \forall (q, r) \in D \tag{2.3}$$

$$x_{uv} = 1 \tag{2.4}$$

$$x_{ij} = \{0, 1\} \qquad \text{for all } (i, j) \in E \tag{2.5}$$

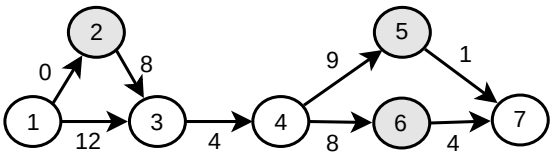

FIGURE 2.3. This Subpath Constrained Shortest Path Problem is not solved by its LP relaxation. The optimal IP solution path with $e = (3, 4)$ and $D = \{(2, 5), (2, 6)\}$ is $1 \to 3 \to 4 \to 5 \to 7$ and has cost 26. In the LP relaxation flow is split between $1 \to 2 \to 3 \to 4 \to 5 \to 7$ and $1 \to 3 \to 4 \to 6 \to 7$ resulting in a lower cost of 25.

The objective is to minimize the cost function (2.1) subject to the flow balance constraints (2.2), the *dominated subpath constraints* (2.3) and the *e in path constraint* (2.4). 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 2.3 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 2.4.

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 2.4 will provide evidence that checking this condition is a highly effective method for verifying that edges are weak.

Theorem 2.1 guarantees that if there is no feasible solution to the Subpath Constrained Shortest Path Problem, then edge e is non-weak. Theorem 2.2 guarantees that if the optimal solution to the IP has a cost greater that U_{st} then edge e is also non-weak.

Theorem 2.1. *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 $K_{sqrt}^{'} \subseteq K_{st}^{'}$ 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 $\forall (q,r) \in D$, $k \notin K_{sqrt}^{'}$. If $K_{st}^{*} = \emptyset$, then e is non-weak.*

Proof. Assume $K_{st}^{*} = \emptyset$. Then $\forall k \in K_{st}^{'}$, $\exists (q,r) \in D$ such that $k \in K_{sqrt}^{'}$. However, by Corollary 2.1 and Lemma 2.2, if $k \in K_{sqrt}^{'}$ for $(q,r) \in D$, then k is suboptimal for all ω. Thus e is non-weak. □

Theorem 2.2. *Consider edge costs $c^{'}$ where all edges in $\overline{SP}_{st}$ have upper bounds costs $\overline{c}$ on and all other edges in $E_{st} \backslash \overline{SP}_{st}$ 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. Assume $K_{st}^{*} \neq \emptyset$ and that $\forall k \in K_{st}^{*}$, $c^{'}(k) > U_{st}$. For every $k \in K_{st}^{*}$ and $\forall \omega$,

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

Therefore

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

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 2.1 for every path $k \in K_{st}^{'} \backslash K_{st}^{*}$, k is suboptimal for all ω. Thus e is non-weak. □

Algorithm 2.1 summarizes our SSPPC Preprocessing Algorithm. We have established correctness for this algorithm in Theorems 2.1 and 2.2. After preprocessing, in

Algorithm 2.1 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 $e=(u,v) \in E_{st} \backslash \overline{SP}_{st}$ **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**
      $D = D \bigcup \{(q,r)\}$
    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

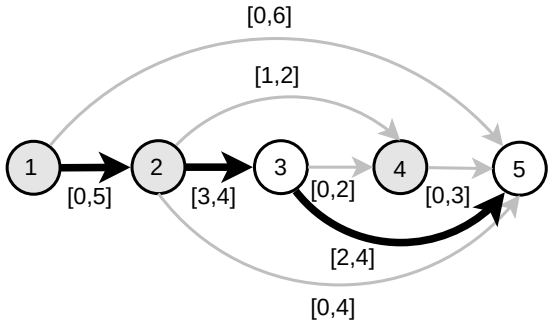

FIGURE 2.4. This figure provides an example where the optimal solution (in bold) to the Subpath Constrained Shortest Path Problem fails to detect that $e = (2, 3)$ is non-weak, where $D = \{(1, 4), (2, 4)\}$.

any given network there may be a small subset of edges that cannot be classified as either being weak or being non-weak. To illustrate where our method fails to detect a non-weak edge, consider the network in Figure 2.4. 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.

2.4. 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 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 can be evidenced in the linear trend in Figure 2.5. Further detail about the running time, the number of IPs and the number of IPs solved by their linear relaxation is summarized in Table 2.1.

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

Stats	Num Edges	Time	Num of IPs	Num 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 2.1. 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.

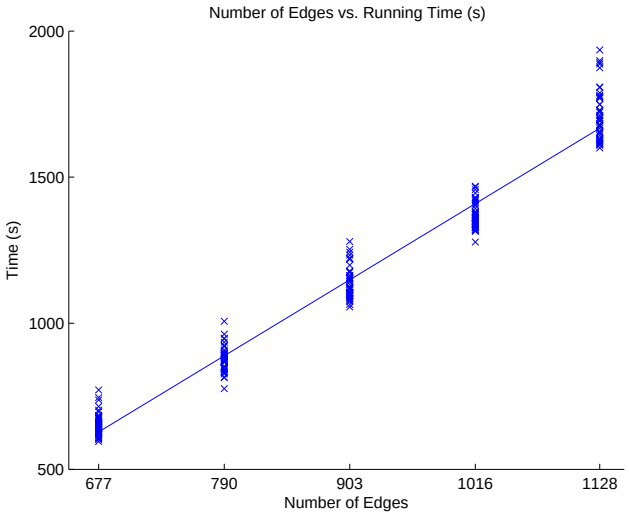

FIGURE 2.5. The running time distribution of 50 tests networks for each edge size is shown above. The linear trend shows that the running time is increasingly at an approximately linear rate in the number of edges for a fixed number of nodes ($|V| = 48$).

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 G_{st} network. Further detail about the computational results for classification of edges is provided in Table 2.2.

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 mod-

Stats	Num Edges	Num After Prepro-cessing	Num Rem-oved w/o IP	Num Rem-oved w/ IP	Num Proved Weak w/o IP	Num Proved Weak w/ IP	Num In UBst	Num Not Class-ified
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 2.2. 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.

eling 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 chapter can be applied to more general stochastic network flow and stochastic optimation problems. Additionally, for specific frameworks within a class of stochastic optimization problems there may be potential to develop more specialized preprocessing methods.

3. Empirical Effects of Clustering on Price Stability in a Stochastic Model for Power Grids.

3.1. Introduction

Operation planning on hydropower systems has received significant attention in the literature in recent years (Keppo 2002, Barros et al. 2003, De Ladurantaye et al. 2007). Stochastic optimization models often help guide decisions about water distribution.

Our contribution in this research is to present and test the hypothesis that clustering can be *effectively* applied to the set of scenarios (or outcomes) in a probability space, mapping that probability space to a scenario tree in which the number of scenarios is dramatically reduced. *Effective* clustering minimizes the loss of information that inevitably results from reducing the size of the probability space.

We investigate this hypothesis in the context of an operations planning problem related to the Brazilian power system. In Brazil, by contract, the prices paid to operators of hydroelectric energy plants are the dual prices of the demand-supply equation from a stochastic optimization model originally developed by Pereira (1989). Consequently the stability of these dual prices is crucial and will be used as a means of assessing the effectiveness of the clustering method we present in this chapter.

Since the Brazilian power system is predominantly hydroelectric, the availability of water is critical for determining optimal prices and ensuring the availability of energy. To represent the stochasticity of the water supply, we use independent scenarios generated by a widely used Periodic Autoregressive model known as the PAR (p) model (Noakes et al. 1985, Maceira and Damázio 2005). By clustering these scenarios, we obtain scenario trees that can be input into the stochastic optimization operation planning model that is used in practice.

Jardim et al. (2001) applied clustering in the context of a hydroelectric system

monthly streamflow model. Their approach was to partition the initial data set into specified numbers of clusters, and then to compare the clustered data points to the original data. Our method utilizes repeated clustering to form a scenario tree and assesses the quality of that tree based upon changes in the results obtained from the stochastic optimization model.

More widespread use of stochastic optimization is limited by three main difficulties: (1) the joint expertise required in both stochastic processes and optimization; (2) demanding data requirements; and (3) computational complexity. The scenario tree is central to all three of these difficulties. Henrion et al. (2007), Heitsch and Römisch (2003), Dupacova et al. (2003), Growe-Kuska et al. (2003) address computational complexity by using probability metrics for scenario reduction. Our clustering method complements this work by providing scenario trees that are already compact and informative from independent time-series scenarios.

While this chapter focuses on operation planning for the Brazilian hydropower system, successful clustering results from sets of scenarios in this context will motivate the application of clustering to sets of scenarios in other stochastic optimization contexts.

The remainder of this chapter is organized as follows. Section 3.2 provides background information on clustering and describes our methodology for generating scenario trees via clustering. Section 3.3 introduces the underlying water distribution problem and the stochastic optimization model we use for the operations planning problem. Section 3.4 summarizes our computational results. Finally, Section 3.5 presents consequences of our work and future research directions.

3.2. Clustering

3.2.1. Background

Clustering has been studied extensively in various research fields such as data mining, machine learning, pattern recognition, and scientific, engineering, social, economic, biological and medical data analysis (Kogan et al. 2006). Clustering is the partitioning of a data set into subsets, so that the data points within each subset are closest to one another, according to some defined distance measure. Euclidean distance is the most commonly used metric (Everitt 1993), but it is not the only metric available for measuring similarity between data sets. Manhattan distance and mahalanobis distance are two notable examples (Jain and Dubes 1988) of other distance metrics that have been used in cluster analysis.

Clustering algorithms are divided into two main branches. In the *hierarchical* branch, successive clusters are identified using previously established clusters. Hierarchical algorithms can be *agglomerative*, where each data point is initially a cluster and larger clusters are formed by grouping together smaller ones. Alternatively, hierarchical algorithms can be divisive, where there is initially a single cluster and smaller clusters are formed by breaking apart larger ones. Examples of hierarchical algorithms include simple linkage, complete linkage, group average, centroid, median and Ward's clustering (Everitt 1993).

The other main branch is partitional clustering algorithms, where a single partition is generated. The $k-$means algorithm is by far the most commonly used partitional algorithm and is perhaps more commonly used than any of the hierarchical methods (Kogan et al. 2006). In $k-$means, the number of subsets k either can be specified initially or determined during the partitioning; and each data point is placed in the subset whose center is nearest.

With recent progress in optimization theory and advances in computational capabilities, optimization-based clustering techniques have begun to emerge (Sherali and

Desai 2005, Tan et al. 2007).

3.2.2. Scenario Trees

Scenario trees are a fundamental component of stochastic programs. They describe the dependence of future information on prior outcomes. Formulations based on scenario trees can enforce *non-anticipativity*, the *here and now* approach to modeling, where decisions must be made while information about the future remains unknown. Decisions made at each stage must therefore hedge against all possible future outcomes. At a later stage, recourse actions are associated with each possible outcome and are based on prior decisions affecting those outcomes. For example, a recourse action taken at a later stage may be to purchase generation capacity from a more expensive source, if at an earlier stage we used most of our available water (the here and now decision) and the rainfall levels (the unknown at the time the decision was made) since that stage were too low to refill the reservoirs.

The basic tradeoff to consider when converting sets of time-series into a scenario tree is: representing enough of the diversity of the time-series data at each stage versus creating a tree that is so large that the problem is computationally intractable. Time-series of forecasted rainfall have heteroscedasticity; the variance of the predicted rainfall increases as we forecast further into the future. So it is reasonable to argue that appropriate trees should have many leaves. At the same time, the most important decisions are those at the first few stages, so we want to ensure that enough diversity is considered early on.

3.2.3. Recursive Algorithm for Generating Scenario Trees via Clustering

Since rainfall is continuous, the probability space containing all possible levels of rainfall would naturally have an uncountable number of scenarios. We approximate this continuous set of scenarios by a finite discrete space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$, where

m is a large integer. In nature, rainfall may vary by season and may have a tendency to follow certain common trajectories.

The time-series data obtained using the PAR (6) model provides rainfall scenarios for r regions from the current time $t = 0$ to some future time $t = n$. As we look further out into a time horizon, our ability to accurately forecast rainfall is decreased. Accordingly, when clustering the PAR (6) time-series data we aim to emphasize proximate data points over distant ones. We accomplish this by first normalizing the time-series and then reweighting them by monotonically decreasing weights. After clustering is complete at a given stage i, the data points for stage i are removed from all the time-series, so that they do not affect the clustering at stages that occur after the time of stage i.

The root node of the scenario tree occurs at $t = 0$. Let $\vec{k}_t$ be the number of nodes desired at time $t = 1, \ldots, n$ of the scenario tree. Let S be an array consisting of the original m time-series scenarios and Q be the corresponding array consisting of the m normalized and then reweighted time-series. At $t = 1$, we apply k-means clustering to the time-series in Q (all data points for $t = 0$ have been removed from Q and S) with $\vec{k}_1$ as the requested number of clusters. The values at the $\vec{k}_1$ nodes in stage 1 are obtained by averaging the $t = 1$ data points of the scenarios within each of the respective clusters. The $t = 1$ data points are then removed from all the time-series in Q and S. Each of the $\vec{k}_1$ clusters enters into stage 2 and the ratio of the number of scenarios in the current cluster to the total number of scenarios m in the original time-series data is calculated. This ratio gives the probability p of the current node v and is then multiplied by $\vec{k}_2$ to determine the number of branches that will be formed at the current step. If more than one branch is required, k-means clustering is then executed and the process continues to stage 3, etc. If no more than one branch is required, then nodes are created for the remaining time steps by averaging the data of the scenarios in the current node for each of those remaining time steps. This process is summarized in Algorithm 3.1.

Algorithm 3.1 Generate a scenario tree from arrays S and Q of m independent time-series scenarios, where S contains the original time-series, Q contains the corresponding normalized and reweighted time-series, $t = 1$ and $v = \emptyset$ initially, and $\vec{k}$ provides a lower bound for the number of nodes to be generated at each stage of the tree.

GenerateScenarioTree $(S,Q,m,\vec{k},t,v)$
  $p =$ (the number of scenarios in $S)/m$
  $\tilde{k} = \text{round}(p \cdot \vec{k}_t)$
  Average the data in the first time index of S to create the current node v' with probability p
  Parent$(v') = v$ and Stage$(v') = t - 1$
  Remove the first time index from S and Q
  if $\tilde{k} > 1$ **then**
    $\{Q_i\}_{i=1}^{\tilde{k}} = k\text{-means}(Q,\tilde{k})$
    Partition S into $\{S_i\}_{i=1}^{\tilde{k}}$ corresponding to $\{Q_i\}_{i=1}^{\tilde{k}}$
    for all i in $1,\ldots,\tilde{k}$ **do**
      GenerateScenarioTree $(S,Q,m,\vec{k},t+1,v')$
    end for
  else
    Average the data in S to create nodes for each of the remaining time steps, all with probability p and assign Parents and Stages to those nodes
  end if

When Algorithm 3.1 terminates, $\vec{k}$ provides a lower bound for the number of nodes at a given stage, rather than an exact number. For example, suppose there are originally 100 scenarios and $\vec{k}_1 = 2$. Let the sizes of the two clusters identified by k-means be 1 and 99 scenarios. Let $\vec{k}_2 = 4$. Since the cluster with 99 scenarios has a ratio of $4(\frac{99}{100})$, those 99 scenarios are partitioned into 4 clusters. However, the 1 scenario remains its own cluster, yielding a total of 5 clusters at stage 2, which is greater than $\vec{k}_2 = 4$. This prevents outliers from being averaged out by the recentering step of k-means clustering. This is significant due to the influence of outliers in stochastic optimization.

Comparing results from stochastic optimization on scenario trees formed through finer (more clusters) and coarser (fewer clusters) clustering will serve as a means of assessing the effectiveness of a given $\vec{k}$ in Algorithm 3.1. If the optimal strategies for distributing water that are obtained under finer clustering differ significantly from the optimal strategies obtained under coarser clustering, that will serve as evidence that finer clustering is needed.

3.3. Operation Planning

The operation planning problem for the Brazilian power system is to determine a strategy for utilizing the available hydroelectric and thermal generation resources, while maintaining the ability to satisfy energy demands over a given time horizon. The Brazilian system can be modeled by four main regions: South, Southeast, North and Northeast. Each of those regions contain aggregated reservoirs and thermal generation plants; and energy can be transferred from one region to another.

To emphasize the difficulty of this problem consider the planning problem for a single region and the greedy strategy of using all the water resources available in the first time period. Suppose there are two options for thermal generation: (1) a capacitated inexpensive option and (2) an uncapacitated premium option. Hydroelectricity

is the least expensive source of energy, but an unexpected drought occurs in the second time period and the capacitated thermal resources are insufficient to satisfy demand. The remaining option is to generate thermal energy at premium prices, but this could have been avoided by implementing a more conservative strategy in the first period. If the capacitated thermal generation had been used in place of some of the water resources, then the remaining water resources could have been used in the second time period and premium prices could have been avoided.

To obtain operation strategies that minimize the expected value of the operation cost during a given time horizon, we will use the following well-known stochastic optimization model (Pereira 1989, Maceira and Damázio 2005). The dual prices from the demand-supply equation are by contract the prices paid to operators of hydroelectric energy plants in Brazil.

3.3.1. Stochastic Optimization Model

Sets:

REGIONS	South, Southeast, North and Northeast
NODES	Nodes in the rainfall scenario tree
ROOT-NODE	The root node
STAGES	The time-horizon for planning

Variables:

GT(NODES, REGIONS)	Thermal generation
GT^{+}(NODES, REGIONS)	Premium thermal generation
V(NODES, REGIONS)	energy storage level
Q(NODES, REGIONS)	Hydroproduction
S(NODES, REGIONS)	Spillage

EV(NODES, REGIONS)	Evaporated energy in reservoirs
F(NODES, REGIONS, REGIONS)	Energy flow between regions
L(NODES, REGIONS)	Loss to excess unstorable supply

Parameters:

β	Time value of money discount factor
δ	Cost penalty for premium thermal generation
γ(REGIONS)	Percentage of water in reservoir that evaporates
fc(REGIONS)	Correction for the effect of head variation
ϵ(REGIONS)	Percentage of rain inflows that are lost
b(NODES)	Parent node
p(NODES)	Node probabilities
t(NODES)	Stages of nodes
$\bar{gt}$(STAGES, REGIONS)	Maximum thermal generation
$\underline{\mathrm{f}}$(STAGES, REGIONS, REGIONS)	Minimum flow between subsystems
$\underline{\mathrm{f}}$(STAGES, REGIONS, REGIONS)	Maximum flow between subsystems
$\underline{\mathrm{q}}$(REGIONS)	Minimum hydroproduction
$\bar{q}$(REGIONS)	Maximum hydroproduction
$\bar{v}$(REGIONS)	Maximum reservoir storage
c(STAGES, REGIONS)	Cost per unit of thermal generation
a(NODES, REGIONS)	Energy inflow from rainfall
d(STAGES, REGIONS)	Energy demand

Constraints:

Storage balance between nodes in each region:

$$V_{n,r} = V_{b(n),r} + \gamma_r \cdot fc_r \cdot a_{b(n),r} - Q_{b(n),r} - S_{b(n),r} - EV_{b(n),r}$$

Demand supply at all nodes in each region:

$$\begin{aligned} d_{t(n),r} &= (1-\gamma_r)\cdot a_{n,r} + Q_{n,r} + GT_{n,r} + GT^{+}_{n,r} + \\ &\quad + \sum_{r'}[F_{n,r',r} - F_{n,r,r'}] - L_{n,r} \end{aligned} \tag{3.1}$$

Bounds on storage for each region:

$$V_{n,r} \leq \bar{v}_r$$

Bounds on hydroproduction for each region:

$$\underline{q}_r - S_{n,r} \leq Q_{n,r} \leq \bar{q}_r$$

Bounds on thermal generation for each region and each time period:

$$GT_{n,r} \leq \bar{gt}_{t(n),r}$$

Bounds on energy flow between regions:

$$\underline{f}_{t(n),r,r'} \leq F_{n,r,r'} \leq \bar{f}_{n,r,r'}$$

Evaporation in reservoirs:

$$EV_{n,r} = \epsilon_r V_{n,r}$$

Final stage storage balance:

$$Q_{n,r} \leq V_{n,r}$$

Objective:

Minimize the expected value of the operation cost during the planning period:

$$\min \sum_{n,r} p(n)\cdot \beta^{1-t(n)} c_{t(n),r}[GT_{n,r} + \delta \cdot GT^{+}_{n,r}] \tag{3.2}$$

3.4. Computational Results

The computational results we have obtained allow us to analyze the impact of two variance factors on the dual-prices of interest: the variability in the clustering process due to (1) randomness in the k-means clustering and (2) the size and structure of the scenario tree. We tested Algorithm 3.1 for generating scenario trees via clustering by generating trees of varying size, stage-depth and structure. For each tree topology, we generated 60 trees using random seeds for the clustering. To construct these trees we used the same $10,000$ samples of 12-month time-series rainfall data from a PAR (6) model.

We implemented the clustering routine described in Section 3.2 in R. We used two k-means routines from the R cluster library, namely Pam and Clara. The Pam routine is more robust but is computationally burdensome for large data sets, so Clara is used for the initial clustering at the root of the scenario tree and Pam is used for later stages. We use a geometric sequence of monotonically decreasing weights with ratio 4/5, *i.e.*, $1, \frac{4}{5}, (\frac{4}{5})^2, \ldots$, to ensure that proximate times for a given stage receive greater weight than distant ones. This ratio should depend on the quality of rainfall forecasting. Choosing the weights to be used is itself an area of active research (Zhao and Zhang 2006).

For all the trees, there is a single root node at $t = 0$. The trees then branch out to $4, 8, 16, 32, 64, 128, 256, 512$ or 1024 nodes at $t = 1$. While all trees contain information through $t = 12$, some trees stop branching at $t = 1$ while others continue to branch out until $t = 3$ and to as many as 1024 leaves. The tree topologies were chosen to study the effects of depth of branching, tree size and tree structure. Throughout the remainder of this section we will use the notation 1-4-1024 to refer to a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$.

We implemented the operations planning model described in Section 3.3 in AMPL and used CPLEX 11.2 as the solver. We obtained data for the remaining model pa-

Tree Structure	Mean	StDev	Min	Max
1-4	379.46	7.82	362.49	391.12
1-8	379.29	5.56	365.33	390.18
1-16	378.05	3.75	369.47	386.09
1-32	376.48	2.94	368.83	382.12
1-64	375.49	2.34	368.92	380.23
1-128	375.63	1.74	370.66	378.23
1-256	375.12	1.33	372.17	378.50
1-512	374.79	0.78	372.90	377.07
1-1024	374.33	0.51	373.17	375.51

TABLE 3.1. Summary of the dual prices for 60 sample trees for each scenario tree topology with 2 stages. Recall, 1-4 is a tree with a single node at time $t = 0$ and 4 nodes at $t = 1, \ldots, 11$.

Tree Structure	Mean	StDev	Min	Max
1-4-256	377.96	2.03	374.01	382.04
1-8-256	377.61	1.73	374.06	381.33
1-16-256	377.11	1.46	373.47	379.92
1-32-256	376.42	1.30	373.84	379.37
1-64-256	376.00	1.16	374.02	379.28
1-128-256	375.53	0.97	372.53	377.46

TABLE 3.2. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages and 256 leaves. Recall, 1-4-256 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 256 nodes at $t = 2, \ldots, 11$.

rameters from historical data available on the Operador Nacional do Sistema Elétrico (ONS) website (http://www.ons.org.br), with the help of Joari Costa from ONS. The dual prices for the demand-supply equation (3.1) provide the optimal pricing strategy for hydropower. These dual prices are a function of costs for thermal production and discount factor. In order to magnify the effects of the discount rate, we introduce a rate of 12% per month. We estimate the costs for thermal production and therefore the true dual prices can be obtained by scaling the our results by a constant.

Tables 3.1, 3.2, 3.3 and 3.4 show the variability in these dual prices for each re-

Tree Structure	Mean	StDev	Min	Max
1-4-1024	377.67	1.69	373.47	380.57
1-8-1024	377.11	1.34	374.60	379.53
1-16-1024	376.53	1.32	373.32	379.54
1-32-1024	376.50	1.04	373.82	378.73
1-64-1024	376.06	0.92	373.24	378.57
1-128-1024	375.61	0.68	373.95	376.73
1-256-1024	375.62	0.70	374.16	377.62
1-512-1024	375.22	0.62	373.53	376.93

TABLE 3.3. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages and 1024 leaves. Recall, 1-4-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$.

Tree Structure	Mean	StDev	Min	Max
1-4-256-1024	379.14	1.91	375.21	383.26
1-8-256-1024	378.54	1.84	373.56	381.45
1-16-256-1024	378.31	1.26	375.72	382.01
1-32-256-1024	377.29	1.23	374.84	379.48
1-64-256-1024	377.08	1.04	373.80	379.23
1-128-256-1024	376.59	1.02	374.27	378.74

TABLE 3.4. Summary of the dual prices for 60 sample trees for each scenario tree topology with 4 stages and 1024 leaves. Recall, 1-4-256-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$, 256 nodes at $t = 2$ and 1024 nodes at $t = 3, \ldots, 11$.

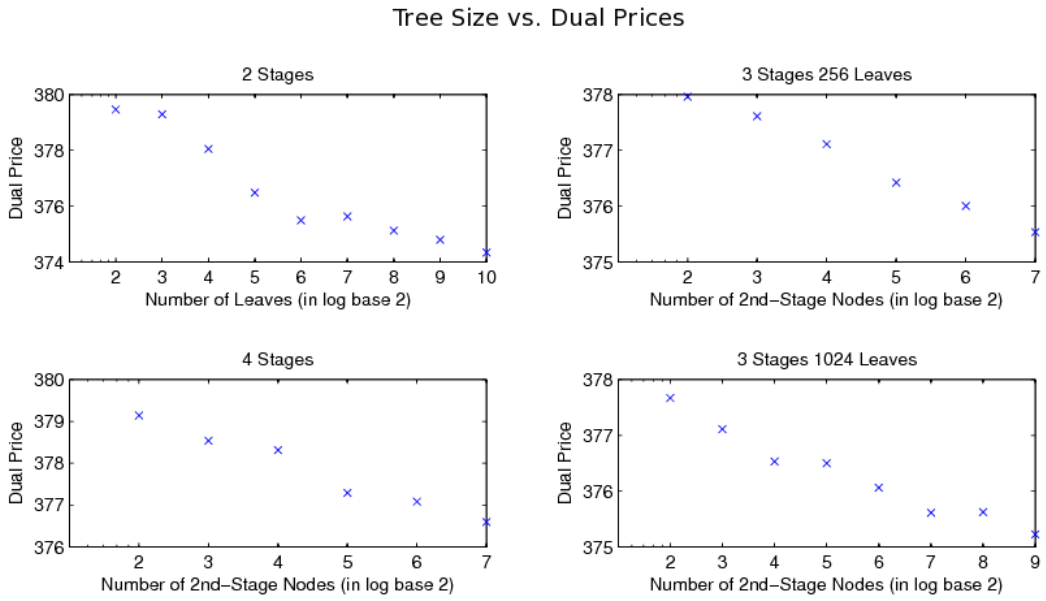

FIGURE 3.1. The four plots above illustrate the relationship between dual prices and tree structure.

quested scenario tree topology. In all four tables, we can see that as the number of nodes (either intermediary nodes or leaves) increases, the standard deviation decreases. Additionally, the following three observations are evidenced in these results:

1. For a fixed number of stages, increasing the number of leaves at the final stage results in lower dual prices. This can be seen in Table 3.1.

2. For a fixed number of stages and leaves, increasing the number of nodes at an intermediary stage results in lower dual prices. This can be seen in Tables 3.2, 3.3 and 3.4.

3. For a fixed number of leaves, increasing the number of stages results in higher dual prices. This can be seen by comparing Tables 3.3 and 3.4.

Figure 3.1 illustrates the trends described in these three observations. Observations 1 and 2 are due to the fact that as the number of nodes and leaves is increased,

the clusters represented in the scenario tree become increasingly more diverse. The addition information obtained through this diversity can be leveraged to delay purchases in many branches until later time periods. For example outlying scenarios that are averaged into larger clusters in smaller trees may be isolated in larger trees, so that earlier purchases may only be required in those isolated branches.

Observation 3 can be similarly explained since as the number of stages is increased for a fixed number of leaves, the information contained in the leaves does not become available until a later time period. Therefore midterm decisions must be made with less information than is available when the same number of nodes are present at an earlier stage.

We have formally verified these observations in two ways: statistically and experimentally. For our statistical analysis, we applied t-tests for the differences in means of two normal distributions with unknown variances that are not necessarily equal. The results of our t-tests provided at least 99% confidence intervals for each of the following comparisons:

- 1-4 has a smaller mean dual price than 1-64. 1-64 has a smaller mean dual price than 1-1024.
- 1-4-256 has a smaller mean dual price than 1-16-256. 1-16-256 has a smaller mean dual price than 1-128-156.
- 1-4-256-1024 has a smaller mean dual price than 1-16-256-1024. 1-16-256-1024 has a smaller mean dual price than 1-128-156-1024.
- 1-4-1024 has a smaller mean dual price than 1-32-1024. 1-32-1024 has a smaller mean dual price than 1-512-1024.

Our data shows that these trends hold between all adjacent pairs, as can be seen both in Figure 3.1 and Tables 3.1, 3.2, 3.3, 3.4. However we did not perform t-tests

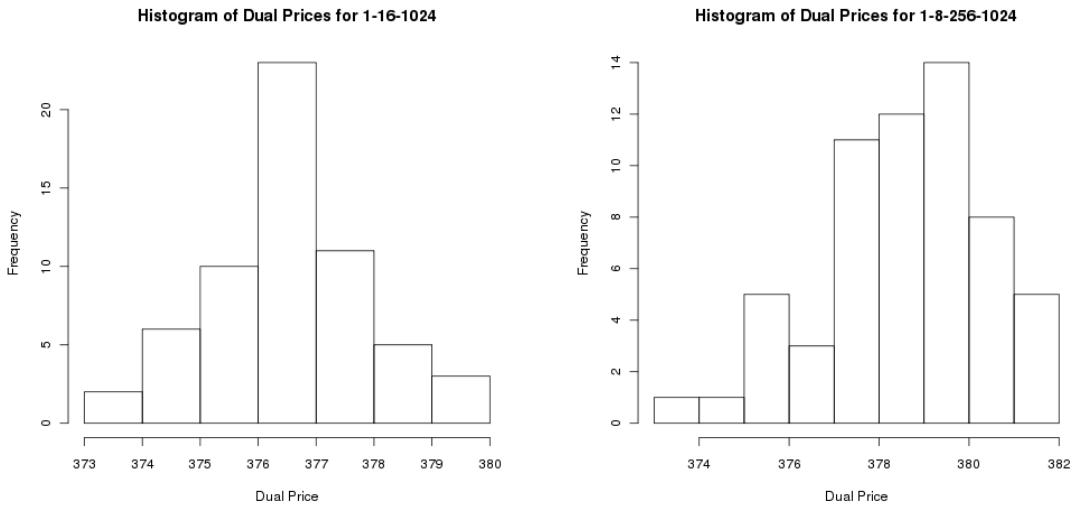

FIGURE 3.2. The histogram on the left provides an example of a tree whose distribution is highly symmetric: 1-16-1024. The histogram on the right provides an example of a tree whose distribution is not symmetric: 1-8-256-1024.

to establish similar confidence intervals for all adjacent pairs since we believe that the observed decay in the differences between means would require significantly larger sample sizes than the 60 that we constructed for each tree structure.

While our distributions were not normal, as we verified through Kolmogorov-Smirnov tests, they are for the most part symmetric and are consistently well-behaved. As a result, our sample sizes of 60 are sufficiently large for obtaining valid conclusions from the t-tests. Figure 3.2 shows the histograms of two distributions: from all the trees tested, 1-16-1024 (a single node at $t = 0$, 16 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$) has one of the most symmetric distributions and 1-8-256-1024 (a single node at $t = 0$, 8 nodes at $t = 1$, 256 nodes at $t = 2$ and 1024 nodes at $t = 3, \ldots, 11$) has one of the least symmetric.

Our second method for verifying observations 1 through 3 was to repeat the optimization component with no discount factor. Removing the monetary incentive to postpone purchases until later time periods resulted in dual prices with significantly less fluctuation than with the original discount factor of 12%. By comparing Tables

Tree Structure	Mean	StDev	Min	Max
1-4	488.84	0.84	487.00	490.10
1-8	488.81.29	0.60	487.32	489.99
1-16	488.67	0.40	487.75	489.54
1-32	488.48	0.32	487.68	489.10
1-64	488.36	0.27	487.66	488.89
1-128	488.30	0.23	487.70	488.69
1-256	488.19	0.21	487.73	488.66
1-512	487.99	0.21	487.33	488.38
1-1024	487.64	0.19	487.32	488.09

TABLE 3.5. Summary of the dual prices for 60 sample trees for each scenario tree topology with 2 stages and no discount factor. Recall, 1-4 is a tree with a single node at time $t = 0$ and 4 nodes at $t = 1, \ldots, 11$.

Tree Structure	Mean	StDev	Min	Max
1-4-256	488.61	0.22	488.08	489.11
1-8-256	488.60	0.22	488.18	489.41
1-16-256	488.57	0.19	488.17	489.32
1-32-256	488.47	0.19	487.91	488.91
1-64-256	488.38	0.17	487.84	488.78
1-128-256	488.31	0.17	487.75	488.58

TABLE 3.6. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages, 256 leaves and no discount factor. Recall, 1-4-256 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 256 nodes at $t = 2, \ldots, 11$.

3.1 and 3.5; Tables 3.2 and 3.6; Tables 3.3 and 3.7; and Tables 3.4 and 3.8, we can verify that the ability to postpone purchases until later time periods is in fact central to the decreasing dual prices obtained through increasing tree sizes.

3.5. Summary and Conclusions

In this chapter, we have introduced a methodology for generating scenario trees from independent samples using k-means clustering. The trees can then be input into a stochastic optimization operation planning model. Our computational results provide

Tree Structure	Mean	StDev	Min	Max
1-4-1024	488.57	0.24	487.80	489.05
1-8-1024	488.53	0.31	487.67	489.26
1-16-1024	488.63	0.29	487.90	489.15
1-32-1024	488.56	0.27	487.97	489.06
1-64-1024	488.59	0.32	487.99	489.48
1-128-1024	488.51	0.19	487.94	488.95
1-256-1024	488.38	0.26	487.65	488.83
1-512-1024	488.22	0.18	487.76	488.56

TABLE 3.7. Summary of the dual prices for 60 sample trees for each scenario tree topology with 3 stages, 1024 leaves and no discount factor. Recall, 1-4-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$ and 1024 nodes at $t = 2, \ldots, 11$.

Tree Structure	Mean	StDev	Min	Max
1-4-256-1024	489.00	0.25	488.32	489.76
1-8-256-1024	488.95	0.21	488.45	489.54
1-16-256-1024	488.96	0.21	488.51	489.35
1-32-256-1024	488.84	0.27	488.07	489.48
1-64-256-1024	488.79	0.20	488.28	489.31
1-128-256-1024	488.68	0.19	488.26	489.02

TABLE 3.8. Summary of the dual prices for 60 sample trees for each scenario tree topology with 4 stages, 1024 leaves and no discount factor. Recall, 1-4-256-1024 is a tree with a single node at time $t = 0$, 4 nodes at $t = 1$, 256 nodes at $t = 2$ and 1024 nodes at $t = 3, \ldots, 11$.

key insights regarding the variability introduced by both the choice of topology and the k-means clustering.

With the best available data from the ONS, we have shown that the dual-prices obtained from the stochastic optimization operation planning model are quite stable for all tree topologies of sufficient size (all topologies with at least 32 clusters in stage 2). As the scenario tree grows, the variability introduced by k-means becomes negligible. Throughout all of our computational results the variability introduced by the choice of topology was limited to under 2% of the dual-price.

In future research, the methodology developed here can be used to measure the sensitivity of the dual prices obtained from trees generated by Algorithm 3.1 to fluctuations in market energy demand. Additionally, our scenario tree generation methodology can be extended to incorporate agglomerative or divisive clustering algorithms, which may provide additional information regarding the dual-prices of interest as well as further insight into the results obtained in this chapter.

The results obtained in this chapter are specific to the operation planning stochastic optimization model that was used, but in future research the methodology we developed may be applied to sets of scenarios in other stochastic optimization contexts.

References

V. G. Adlakha. An improved conditional Monte Carlo technique for the stochastic shortest path problem. *Management Science*, 32(10):1360–1367, 1986.

J. F. Bard and J. E. Bennett. Arc reduction and path preference in stochastic acyclic networks. *Management Science*, 37(2):198–215, 1991.

J. F. Bard and J. L. Miller. Probabilistic shortest path problems with budgetary constraints. *Computers and Operations Research*, 16(2):145–159, 1989.

M. T. L. Barros, F. T. C. Tsai, S. Yang, J. E. G. Lopes, and W. W. G. Y. H. M. Asce. Optimization of large-scale hydropower system operations. *Journal of Water Resources Planning and Management*, 129:178–188, 2003.

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.

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.

S. Chanas and P. Zieliński. On the hardness of evaluating criticality of activities in a planar network with duration intervals. *Operations Research Letters*, 31(1):53–59, 2003.

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.

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.

W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Inc. New York, NY, USA, 1998.

D. De Ladurantaye, M. Gendreau, and J. Y. Potvin. Optimizing profits from hydroelectricity production. *Computers and Operations Research*, 2007.

J. Dupacova, N. Growe-Kuska, and W. Romisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.

A. Eiger, P. B. Mirchandani, and H. Soroush. Path preferences and optimal paths in probabilistic networks. *Transportation Science*, 19(1):75–84, 1985.

B. S. Everitt. *Cluster Analysis*. Edward Arnold, London, England, 1993.

P. C. Fishburn. Utility theory. *Management Science*, 14(5):335–378, 1968.

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.

H. Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17(4):583–599, 1969.

N. Growe-Kuska, H. Heitsch, and W. Romisch. Scenario reduction and scenario tree construction for power management problems. In *Power Tech Conference Proceedings, 2003 IEEE Bologna*, volume 3, 2003.

H. Heitsch and W. Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2):187–206, 2003.

R. Henrion, C. Küchler, and W. Römisch. Scenario reduction in stochastic programming with respect to discrepancy distances. *Computational Optimization and Applications*, pages 1–27, 2007.

Y. C. Ho, R. S. Sreenivas, and P. Vakili. Ordinal optimization of DEDS. *Discrete Event Dynamic Systems*, 2(1):61–88, 1992.

A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

D. Jardim, M. E. P. Maceira, and D. M. Falcao. Stochastic streamflow model for hydroelectric systems using clustering techniques. In *2001 IEEE Porto Power Tech Conference Proceedings*, volume 3, 2001.

O. E. Karasan, M. C. Pinar, and H. Yaman. The robust shortest path problem with interval data. Technical report, Bilkent University, Department of Industrial Engineering, August 2001.

C. J. Karbowicz and J. M. Smith. A k-shortest paths routing heuristic for stochastic network evacuation models. *Engineering Optimization*, 7(4):253–280, 1984.

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.

J. Keppo. Optimality with hydropower system. *Power Systems, IEEE Transactions on*, 17 (3):583–589, 2002.

B. Klinz and G. J. Woeginger. Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43(3):153–162, 2004.

J. Kogan, C. Nicholas, and M. Teboulle. *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer-Verlag New York, Inc. Secaucus, NJ, 2006.

R. P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. *Communications of the ACM*, 26(9):670–676, 1983.

M. E. P. Maceira and J. M. Damázio. Use of the par(p) model in the stochastic dual dynamic programming optimization scheme used in the operation planning of the Brazilian hydropower system. *Probability in the Engineering and Informational Sciences*, 20 (01):143–156, 2005.

E. Miller-Hooks. Adaptive least-expected time paths in stochastic, time-varying transportation and data networks. *Networks*, 37(1):35–52, 2001.

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.

D. J. Noakes, A. I. McLeod, and K. W. Hipel. Forecasting monthly riverflow time series. *International Journal of Forecasting*, 1:179–190, 1985.

M. V. F. Pereira. Optimal stochastic operations scheduling of large hydroelectric systems. *International Journal of Electrical Power & Energy Systems*, 11(3):161–169, 1989.

H.D. Sherali and J. Desai. A global optimization RLT-based approach for solving the hard clustering problem. *Journal of Global Optimization*, 32(2):281–306, 2005.

J. B. Sidney. The two-machine maximum flow time problem with series parallel precedence relations. *Operations Research*, 27(4):782–791, 1979.

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.

C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg. The stochastic shortest route problem. *Operations Research*, 28(5):1122–1129, 1980.

M. P. Tan, James R. Broach, and Christodoulos A. Floudas. A novel clustering approach and prediction of optimal number of clusters: global optimum search with enhanced positioning. *Journal of Global Optimization*, 39(3):323–346, 2007.

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.

J. A. Ward. Minimum-aggregate-concave-cost multicommodity flows in strong series-parallel networks. *Mathematics of Operations Research*, 24:106–129, 1999.

G. Yu and J. Yang. On the robust shortest path problem. *Computers and Operations Research*, 25(6):457–468, 1998.

Y. Zhao and S. Zhang. Generalized dimension-reduction framework for recent-biased time series analysis. *IEEE Transactions on Knowledge and Data Engineering*, pages 231–244, 2006.

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.