# Reliable Routing with Recourse in Stochastic, Time-Dependent Transportation Networks

by

Stephen David Boyles, B.S.

Thesis

Presented to the Faculty of the Graduate School

of The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2006

Reliable Routing with Recourse in Stochastic, Time-Dependent

Transportation Networks

APPROVED BY

SUPERVISING COMMITTEE:

_____

S. Travis Waller

_____

Jorge Prozzi

# Acknowledgements

This thesis, and my graduate study at The University of Texas at Austin, have benefited greatly from the assistance of many people. My advisor, Dr. S. Travis Waller, not only introduced me to the wonderful world of network algorithms, but also convinced me to pursue a doctorate degree instead of entering the workforce after the master's degree. In the last two years I've also realized that I have a passion for teaching, and I thank Dr. Waller and Dr. Randy Machemehl for offering me opportunities to discover this. I also appreciate the help of Dr. Jorge Prozzi, who gave many useful comments on this thesis.

I also thank each of the members of Dr. Waller's research group, without whom my graduate study would have fallen far short of the recommended allowance of random websites, Photoshopping, and friendly jokes regarding a certain transportation professor at another school, in addition to useful advice on research and personal matters.

Furthermore, I greatly appreciate the support of University United Methodist Church, where I have found a welcoming and deeply satisfying community; and of my parents, who, throughout my entire life, have done more for me than I can express in words.

Thank you all.

Reliable Routing with Recourse in Stochastic, Time-Dependent Transportation Networks

by

Stephen David Boyles, M.S.E.

The University of Texas at Austin, 2006

SUPERVISOR: S. Travis Waller

A general algorithm is presented for finding an optimal routing policy in a stochastic, time-varying network where a user's preferences regarding arrival time are described by a piecewise polynomial disutility function. Additionally, *deviance*, an important disutility function of this class, is introduced. Proofs of correctness and complexity results are derived, as are methods to account for complications in networks with cycles. Numerical investigations consider the required computation time and the impact of using nonlinear disutility functions. These investigations show that large increases in reliability may be attainable at only a slight cost in expected travel time, indicating that correctly accounting for traveler's preferences regarding reliability is crucial.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1: Introduction

## 1.1 Background

The problem of describing the route chosen by a traveler is fundamental to the study of transportation networks. On the scale of individual vehicles, finding a route that is "best" in some sense (for instance, quickest or cheapest) is of economic interest to firms which make extensive use of transportation infrastructure, such as shipping or delivery companies. On a macroscopic level, describing the behavior of a large number of users of the system is essential to the urban planning process. This is often described as "traffic assignment," the final step of the traditional four-step planning process.

The most common assumption is that users of the transportation network choose a route which minimizes the cost of travel. For simplicity in solving large-scale problems, travel time is often used as a proxy for cost when considering travel of private citizens, who do not typically consider monetary cost of daily travel, but who implicitly place value on time spent on travel. Additionally, direct measurement of travel time is easier to measure, and allows for modeling simplifications, as any two users on the same route at the same

time will experience the same travel time, whereas their monetary value of that same time may differ.

Thus, in solving the traffic assignment problem, a common approach is to iteratively find and update the paths used by each traveler in the network, until no traveler can decrease their experienced travel time by unilaterally switching routes, at which point we stop and describe the network as being in a user equilibrium (UE) state, which is assumed to characterize behavior in the network. Finding the best path for a single traveler is the basis for finding an equilibrium solution; in UE, the assumption is that the best path has least cost.

Different behavioral assumptions lead to different solutions to the traffic assignment problem: the UE state, described above, occurs when travelers minimize their individual travel times. If travelers instead minimize their marginal contribution to the total travel time experienced by all users, the network is said to have a system optimal (SO) state, where the average cost of all users is minimized. Intuitively, this models "altruistic" behavior, in contrast to the "greedy" behavior of UE where each individual's travel time is minimized. These different models can lead to very different network conditions; thus, we see that the correct specification of individual users' behavior can have significant impact on the network conditions.

In real traffic networks, travel times cannot be characterized by a single, deterministically known value; instead, there is inherent uncertainty in travel times due to congestion, weather, incidents, and other factors. In the face of such uncertainty, the assumption that travelers minimize their individual costs is ill-posed, as the total travel cost is itself uncertain. A common modeling technique is to assume a probability distribution for travel costs, and seek a route which minimizes the expected cost from this distribution; however, this implicitly assumes that travelers are risk-neutral.

In this thesis, this assumption is generalized to consider the case when travelers may be risk-averse or risk-prone. The problem of identifying the optimal route for an individual traveler under these new behavioral assumptions is addressed, and algorithms are developed to solve the problem in general. Although the traffic assignment problem under this assumption is not addressed here, the algorithms developed here are essential to its solution in the future.

## 1.2 Motivation

Where uncertainty and randomness exist in traffic networks, it is necessary to choose an objective that accommodates this stochasticity, such as choosing to minimize expected travel cost. While such an approach has the

advantage of simplicity, it is unable to model more sophisticated descriptions of risk that characterize human behavior.

For example, consider the simple network below (Figure 1.1), where a traveler is attempting to move from node 1 to node 2. Two routes are available, denoted by arcs A and B. Arc A either has cost 5 or 25, with equal probability, while Arc B deterministically has a cost of 16.

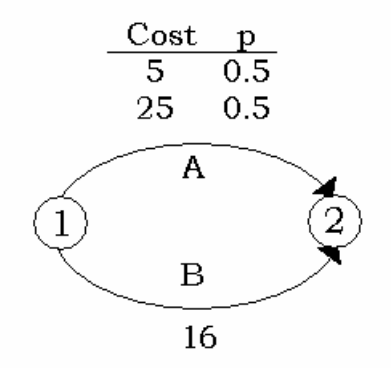


**Figure 1.1**. Motivating risk-averse behavior in transportation networks.

Clearly, the only choice that can be made is whether to take arc A or arc B. As the expected travel costs on the two arcs are 15 and 16, respectively, one ought to choose arc A if minimizing expected cost is the objective. However, half of the time arc A will have a substantially higher cost than this average figure; and in many cases choosing arc B is advantageous: although arc A has a lower cost on average, the cost of arc B is known with complete accuracy, and is only

slightly higher on average. In such a case, the traveler may be willing to suffer a small increase in average travel time in exchange for obtaining reliability.

One might expect this type of behavior in several situations. A commuter traveling to work is likely to behave in a risk averse manner, as the penalty for arriving late generally exceeds the benefit from arriving early. Thus more reliable routes may be more useful than riskier routes, even some that may have a lower cost on average. Another case where such behavior might arise is in supply chains or the delivery of perishable goods. Again, this is a situation where a late arrival is far more costly than an early arrival is beneficial. In such cases it is inappropriate to simply minimize the expected travel time, where any penalty due to a late arrival can be offset by a sufficiently early arrival. Therefore, in this thesis we seek a method of finding paths that accounts for a more general specification of a traveler's preferences regarding arrival times.

Also, with the growing prevalence of intelligent transportation systems (ITS) infrastructure that allows users to learn information about network conditions as they travel (for instance, from observing travel times on a variable message sign (VMS) or through an in-vehicle device), it is useful for transportation models to be able to account for the ability of users to update their routing decisions using information learned *en route*. For instance, if a traveler

receives information indicating that his or her intended route has a much higher cost than anticipated (such as in the case of a severe incident on that route), one would expect the traveler to choose a different route, even if the information was learned while the trip was underway. Consider the network in Figure 1.2, where the arc costs are indicated on the graph. Arcs B and C take either cost 1 or 3, with equal probability, independent of the cost of the other.

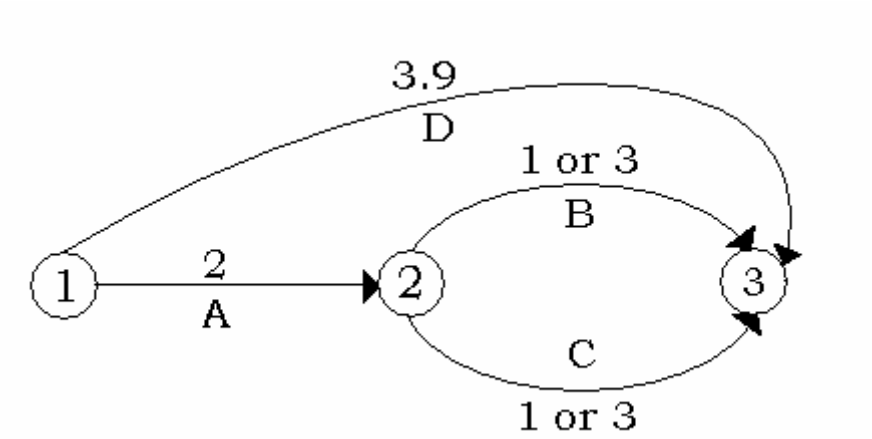


**Figure 1.2**. The importance of accounting for recourse.

This network has three paths: AB, AC, and D. If a traveler did not anticipate receieving any information about the cost of B or C before choosing those nodes, the path with least expected cost is D (3.9 compared to 4). However, now assume that if the traveler arrives at node 2, the costs of B and C will both be learned, perhaps through a VMS device. In this case, the traveler can choose the

arc with lesser cost, if they differ. By doing this, the traveler can guarantee a total cost of 2 + 1 = 3, unless both B and C have cost 3, in which case the total cost will be 5. The probability of both B and C having high cost is 1/4; thus, the expected cost of choosing arc A, then picking the lesser of B and C, is 3.5. This is better than simply following arc D all of the time, which has cost 3.9. It is important to note that even though information is only provided at node 2, this impacts the traveler's decision at the origin. Following arc A is only preferable because the traveler anticipates receieving information at node 2. This implies that it is crucial to consider the impact of information on the entire travel decision. Providing information does not simply cause a reaction at the time and place where this information was given; rather, it can fundamentally alter behavior from the moment of departure, because, as in Figure 1.2, expecting to receive information along a path can make it more attractive.

Indeed, it is not unreasonable to assume that travelers behave this way even in the absence of ITS devices. Although the costs of arcs is uncertain, one expects some dependency between these costs: if a particular freeway segment is heavily congested, it is likely that nearby freeway segments are also congested. Thus even if a traveler only observes the condition of the arcs as they are

traversed, the mere act of traveling through the network provides information on the state of the remaining arcs.

To account for this, the models proposed in this thesis allow a traveler to update the route choice throughout the trip. Methodologically, this means that instead of seeking a single best path, we seek a set of paths, any one of which may or may not be used depending on the experienced network conditions. Allowing this behavior may significantly alter the optimal routing decision, as a traveler may favor routes which allow for more contingent options downstream, anticipating that information will be received to allow a flexible choice of the best route at a later point in time.

Thus, the motivation of this work is twofold: we seek a routing model that allows for a general accommodation of risk regarding uncertain travel costs, and that also allows for the routing decision to be altered *en route* as travelers take advantage of information learned regarding the future state of the network. In doing so, we account for uncertainty in traffic networks in two ways: we model both how uncertainty affects a user's preference for a particular route, and how drivers should react to information received *en route* regarding this uncertainty.

## 1.3 Problem Statement

The aim of this work is to develop the means to find an online routing strategy that is optimal for a traveler whose preferences (utility) can be expressed by a polynomial utility function of travel time, possibly composed of several distinct pieces. We first discuss the nature of this utility function, followed by a description of the travel time distributions we assume. This section concludes with a specification of the type of routing strategy we seek.

As we assume that travel is typically viewed as burdensome, for ease we consider the *disutility* of a trip; that is, the function we choose to describe a traveler's preferences reflects the burden associated with a trip of a given length. Thus, a trip with a lower disutility is preferable. We assume that the traveler wishes to minimize the expected value of this disutility, and seeks a routing policy accordingly.

A piecewise polynomial disutility function is very flexible and can accommodate a wide variety of traveler behavior, including specification of risk preferences and time constraints. In fact, much previous work in online routing under uncertainty can be viewed as a special case of minimizing disutility described by a piecewise polynomial function.

We assume that travel disutility is a function only of travel time. Other factors that might affect disutility, such as monetary tolls, or a desire to avoid freeways, are not considered. Accordingly, the terms "cost" and "travel time" will be used interchangeably in this thesis.

Since we are only considering the decision of one traveler, we assume that the decisions of all other users of the network are fixed; that is, although the travel times are uncertain, we need not consider the impact of our traveler on the route choice of others: the probability distributions governing travel costs are unaffected by the routing policy we seek. Although this significantly eases the difficulty in generating such a policy, this means that additional work must be done to see the macroscopic impacts of modeling traveler behavior in this way.

Also, it is necessary to make some assumptions on the distributions of travel times on links in the network. First, we assume that travel times are integral and that the support of the travel time distribution on any link is finite. This assumption is not limiting, as one can choose units of time sufficiently small to describe costs at any desired precision. Additionally, at any resolution, one may choose as many support points as is necessary to approximate a continuous distribution. We refer to each possible cost as a "state" of that arc.

It is also necessary to specify the type of dependency we will consider. In this thesis, we allow arc costs to contain both limited spatial and temporal dependency. Spatial dependency implies that the state of an arc is influenced by the state of arcs that preceded it on the path. This allows congestion-related phenomena to be modeled: if one arc is highly congested, one might expect the adjacent arcs to also exhibit congestion. In this thesis we assume that this dependency only extends to the previous arc; that is, the distribution of costs on any given arc is completely determined by the state of the previous arc traversed. This is known as one-step spatial dependency, or alternately, Markovian cost structure. Relaxing this assumption to account for multiple-step dependency is straightforward, but requires more cumbersome notation and a substantial increase in algorithmic complexity; for these reasons, here we limit our attention to one-step dependency.

Temporal dependency has several distinct meanings in the literature; in this work, we take this to mean that the probability distributions for arc costs depend on the time at which the arc is traversed. Clearly, in real traffic networks the travel time on an arc is highly dependent on the time of day at which one travels. Thus we propose to model both the temporal and spatial characteristics of arc cost dependencies by specifying, for each arc, a probability distribution

determined by the state of the previous arc traveled, and by the time at which one reaches this arc.

Finally, we assume that the cost distributions on all arcs, taking account of spatial and temporal dependency, are known to the traveler before departing.

With this dependency structure and the piecewise polynomial disutility function in mind, attention can be turned to the nature of the strategy we prescribe for a traveler. Offline, or *a priori*, routing models specify a complete path from the origin to the destination, while online, or recourse, models allow travelers to update this path *en route* and choose a new route. In particular, we allow travelers to update their decision as they learn information which allows them to take advantage of the spatial and temporal dependencies which exist in the cost structure.

Since the cost distribution of any arc conditions on the state on the previous arc, and on the current time, we allow the traveler's decisions to condition on the same information: at any node in the network, the traveler chooses an outgoing arc based on the time and on the state of the last arc that was traversed. Note that no decision is made about any future course of action: at every node (including the origin), the only decision made is about the next arc to follow. Upon arriving at the downstream node, the traveler chooses the next

arc, and so on. Although such a strategy might appear myopic or inefficient, it is actually more flexible and general than complete specification of a path *a priori*: if we prescribe the same set of decisions regardless of any information learned, the result will be a path that is followed deterministically. Since the set of offline routing policies is a subset of the set of online policies, the optimal online policy can be no worse than the optimal offline policy (Waller and Ziliaskopoulos, 2002).

A complete routing policy prescribes a course of action at every node for every possible combination of arrival time and predecessor state. We evaluate strategies according to the specified disutility function. As uncertainty exists both in the network costs and in the path followed (since the eventual path to the destination depends on the stochastic network costs), for any strategy there is an associated distribution of travel times from the origin to the destination. The expected disutility of a strategy is then calculated with respect to this distribution. We seek an efficient method of determining an optimal strategy according to this criterion.

## 1.4 Contributions

To our knowledge, this work is the first to explore online routing behavior in stochastic, time-dependent networks under a general class of disutility functions. Although the literature contains several instances of recourse models in stochastic, time-dependent networks using specific types of disutility functions, this work is the first to treat the problem in greater generality. Likewise, although there has been some research in routing with nonlinear disutility functions, this has exclusively considered networks where arc costs are static and deterministic, and where travelers cannot exercise any recourse options. By combining the two, we produce a more general model. Further, the nature of recourse models allows more efficient and direct methods of solution than has been typically found in the existing literature on routing with a general utility function.

We expect that this model has the potential to model driver route choice behavior more accurately, by accounting for a wider range of behavior, such as the ability to adapt to revealed network conditions; by accounting for the spatial and temporal dependencies in arc cost which exist in reality; and by accounting for a wide range of user preferences and attitudes regarding travel time and risk. No existing model incorporates all three of these factors.

Further, by modeling the behavior of individuals more accurately, the potential exists to model the collective behavior of a large number of users more accurately. This would lead to benefits in the travel planning process, as decisions and policies could be evaluated with a better model for route choice and traffic assignment.

Finally, this work may motivate additional research in calibrating the utility functions that describe user behavior, or in calibrating the probability distributions that represent arc costs, as real-world implementation of such models should incorporate accurate descriptions of these input data.

## 1.5 Organization

Chapter 2 surveys relevant literature in recourse routing, as well as various methods of accounting for a traveler's presumed aversion to risk regarding uncertain travel costs. Chapter 3 develops a model and algorithm to find an optimal policy using one particular measure of disutility, which we term "deviance". Although the problem will be treated in full generality later, many of the concepts that are involved are best illustrated with a specific example. As deviance is an important and useful disutility function, it was chosen for this example. Chapter 4 generalizes the results of the previous chapter, stating an

efficient algorithm for finding the optimal strategy for an arbitrary piecewise polynomial disutility function, and provides proofs of correctness and complexity results. Chapter 5 examines the computational and numerical performance of these algorithms, and Chapter 6 summarizes the contributions of this work and discusses possible extensions of the research described here.

# Chapter 2: Literature Review

## 2.1 Introduction

While this research is the first to examine the general problem of finding an optimal recourse routing strategy with regard to a specified utility function, some previous work exists with regard to specific instances of this problem. Further, there is a considerable body of literature on routing problems which underlies the contributions made in this thesis. Thus, this chapter describes existing research which is relevant to the problem at hand.

For purposes of organization, these are divided into three categories which are discussed in turn. We first discuss research on routing problems in general, followed by various methods to account for uncertainty and travelers' risk preferences, and finally a description of routing models incorporating recourse. This chapter concludes with a brief summary of the state of the art to provide context for the contributions of this thesis.

## 2.2 Routing Problems

The quintessential algorithm for finding optimal routes on mathematical graphs was developed by Dijkstra (1959), and finds the least-cost path between a

pair of nodes in a network with static, deterministic arc costs. The efficient solution of this problem relies on the additivity of path costs; that is, that the cost of a path is the sum of the cost of the individual arcs in that path. This allows an optimal path to be constructed one arc at a time, finding the optimal path from the origin to all nodes, including the destination. At any point we only need to consider adding an arc to one of the shortest paths to some node in the network, since the optimal path cannot contain any suboptimal path. This property, known as Bellman's Principle (1957), is used extensively in routing algorithms.

Dijkstra's algorithm cannot be applied when arc costs may be negative; instead, label correcting approaches, a more general class of algorithms, are employed. These approaches, first developed by Ford (1956), work backwards from the destination one node at a time, constructing paths using Bellman's Principle in the same manner as Dijkstra's algorithm; however, the key distinction is that none of these paths are considered optimal until the algorithm terminates, and can be altered if a better path is found. In contrast, Dijkstra's algorithm never changes a path once it sets one; thus, it is known as a label setting approach. A more detailed description of label setting and label correcting approaches, and the differences between them, can be found in Ahuja et al. (1993).

Label correcting approaches are also commonly used in shortest path routing problems on networks with a more sophisticated arc cost structure; for instance, when arc costs are time-dependent but deterministic (Ziliaskopolous and Mahmassani, 1993), although label setting approaches also exist for this problem (Dreyfus, 1969). Conversely, when arc costs are stochastic, but not temporally or spatially dependent, the path with least expected cost can be found by replacing each arc cost with its expected value and applying a static shortest path algorithm, such as the ones mentioned above.

Hall (1986) proved that extension to the case when costs are both stochastic and time-dependent is not straightforward. Indeed, Fu and Rilett (1998) demonstrate that the minimum computation time to solve this problem is exponential in the worst case, and propose a heuristic to solve the problem. Miller-Hooks and Mahmassani (2000) and Hall (1986) provide exact, nonpolynomial algorithms for this case.

Some authors develop models which do not assume that a traveler's utility is simply the experienced travel time, but instead, a more specific utility functions describes a user's preferences, when the burden of travel is not directly proportional to the duration of travel. Gabriel and Bernstein (1997) discuss a traffic assignment problem with non-additive path costs, and Gabriel and

Bernstein (1999) present an iterative pruning procedure for finding the non-additive shortest path. Tsaggouris and Zaroliagis (2004) present such an algorithm for monotone and convex disutility functions.

However, these models generally assume that disutility increases with travel time. This may not always be true; for instance, arriving much earlier than expected might cause the traveler to feel as though time has been wasted, as his or her departure could have been postponed while still allowing on-time arrival. Indeed, Redmond and Mokhtarian (2001) indicate that a limited amount of commuting time may actually have positive utility: commuters find value in time spent alone while traveling, to transition between the work and home environments. In such cases, an extremely short commute is in fact less desirable than one of moderate length.

## 2.3 Robust Routing Approaches

The value of controlling uncertainty in stochastic networks has long been recognized in the literature. Frank (1969) and Mirchandani (1976) investigate the probability distribution of the cost of the shortest path in networks with stochastic arc costs, while Sigal et al. (1980) examine the probability that a particular path is the shortest. Likewise, the usefulness of modeling user

preferences by a utility function, rather than assuming risk neutrality, has been noted. Loui (1983) and Eiger et al. (1985) develop procedures for linear and exponential utility functions which allow an efficient dynamic programming-like solution, while Murthy and Sarkar (1996) present an algorithm for decreasing quadratic utility functions.

However, a variety of other methods of controlling uncertainty exist as well. Sivakumar and Batta (1994) solve a shortest path problem with a constraint on the maximum allowable travel time variance. Gao (2005) presents a heuristic for finding an online path with minimum variance. Robust formulations, such as that in Yu and Yang (1998) or Montemanni and Gambardella (2004), solve a minimax shortest path problem to provide a reasonable risk-averse solution even when no information on the distribution of arc costs is known, except for the support interval. Multiobjective optimization approaches are also common. For instance, Sen et al. (2001) develop such an approach when arc costs are normally distributed and one wishes to minimize both the mean and variance of travel time. Fan et al. (2005), on the other hand, find a routing policy that minimizes the probability of arriving at the destination later than a specified target arrival time. Another approach involving a desired arrival time is presented in Gao

(2005), where a weighted sum of expected travel time before and after this target, and the expected travel time itself, is minimized.

## 2.4 Online Routing Approaches

More recently, some variants of routing problems with dependent arc costs seek online policies where vehicles can update their path choice en route as they learn information by traversing part of the network, using knowledge of arc cost dependency to improve their expected cost. This is in contrast to offline models where vehicles will not deviate from the a priori optimal path, regardless of any information that is learned *en route*.

By definition, the solution to such problems is not a single path, but rather a set of paths that may be chosen depending on network conditions. The rule for choosing the appropriate path, and updating this choice as information becomes available, is alternately called a routing strategy or routing policy by various authors. Waller and Ziliaskopolous (2002) prove that allowing recourse can only reduce a user's expected cost, compared to the best shortest path *a priori*.

Andreatta and Romeo (1988) develop a model where arcs fail randomly; if a traveler encounters such a failure, a fixed recourse path is followed to the destination. Psaraftis and Tsitsiklis (1993) develop an algorithm that finds a

policy allowing recourse decisions at all nodes, in acyclic networks. Waller and Ziliaskopolous (2002) present an algorithm for cyclic networks with limited temporal and spatial dependency, and Provan (2003) develops a Dijkstra-like procedure for a more general dependency structure with nonnegative arc costs. Miller-Hooks (2001), on the other hand, develops a polynomial algorithm for online routing in stochastic, time-dependent networks with independent arc costs.

The case of general dependency is more difficult. Polychronopolous and Tsitsiklis (1996), and Provan (2003), prove that the problem of finding the optimal strategy in such networks is NP-complete, regardless of the presence or absence of cycles, unless one takes the so-called "reset assumption." Under the reset assumption, repeated visits to an arc are independent stochastic trials; that is, if an arc is visited twice, it is not required to have the same cost on both traversals. Instead, each time an arc is visited, the cost is probabilistically determined from the appropriate distribution.

Gao (2005) and Gao and Chabini (2006) develop a taxonomy for recourse problems in stochastic, time-dependent networks, classifying problems according to the nature of the dependency structure and of the information receieved by travelers.

## 2.5 The State of the Art

While the previous sections describe some of the extensive research made on accounting for uncertainty in routing, and on accounting for the possibility of recourse decisions, relatively little work has been done on combining the two. Gao (2005) presents a heuristic to find a policy with minimum variance, and an exact algorithm for minimizing the weighted sum of three linear penalties for expected travel time and expected arrival before and after a target arrival time, but to our knowledge, this is the only model which attempts to unify these concepts in stochastic, time-dependent, possibly cyclic networks where recourse decisions can be made at all nodes. Additionally, while the literature contains several instances of routing models incorporating utility functions to accommodate non-additive costs, very little research exists to model routing behavior when disutility is not a monotone function of travel time, as Redmond and Mokhtarian (2001) indicate may be a better model of commuter preferences.

This thesis attempts to advance the state of the art by combining the above concepts into a more general routing model: we propose a routing model with recourse, where user preferences can be specified by an arbitrary piecewise polynomial function which need not be monotone, convex, or continuous. Such

a model can incorporate a wide variety of preferences, including risk-averse, risk-prone, and risk-neutral behavior, as well as the possibility that, in some cases, a moderate amount travel time need not be burdensome at all, but actually preferable to an extremely short trip.

# Chapter 3: A Specific Example: Algorithm MDR-MC

## 3.1 Introduction

Before presenting a general formulation and solution algorithm for the problem of finding an optimal policy for a piecewise polynomial utility function, it is useful and instructive to develop such an algorithm for a particular disutility function from first principles. Many of the concepts involved are best illustrated with a specific example, which should improve the clarity both of the algorithm presentation and of various proofs of correctness and complexity.

This chapter develops a model for a specific type of disutility function which we term *deviance*, defined below. In addition to easing the presentation of the general algorithm, deviance is itself an important measure of risk and worthy of mention. The general algorithm is fully described in Chapter 4.

Section 3.2 defines the network topology formally and introduces notation that will be used both in this chapter and in Chapter 4. The nature of the disutility function, and why deviance was chosen, is discussed in Section 3.3. Section 3.4 presents algorithm MDR-MC to solve this problem, along with proofs of correctness and complexity. Section 3.5 discusses the selection of the target arrival time, which is a parameter of deviance, and Section 3.6 introduces a

method to use deviance to only penalize late arrivals, instead of both early and late arrival. Finally, Section 3.7 summarizes the content of this chapter.

## 3.2 Model and Notation

In this thesis we consider networks with Markovian arc costs; formally, let $G = (N, A, \mathbf{P})$ be a probabilistic directed network with finite node and arc sets $N$ and $A$; let $|N| = n$ and $|A| = m$. $\mathbf{P}$ is a rank-5 tensor, described below, that specifies the probability distributions for arc costs. We consider a single origin $O \in N$ and a single destination $D \in N$. Assume that a directed path from each node to $D$ exists. For each arc $a = (a_1, a_2) \in A$ there is an associated set of integer costs, dependent on departure time t, which we assume to be finite, that is, $C_{a,t} = \{c_{a,t}^1, c_{a,t}^2, \ldots, c_{a,t}^{S(a,t)}\}$ where $S(a,t)$ is the number of states associated with arc $a$ at time $t$. Let $M = \max \{c_{a,t}^r : a \in A, t \in \mathbb{Z}^+, r \in \{1, 2, \ldots, S(a,t)\}\}$. In this work we assume arc costs are travel times, and the terms "cost" and "travel time" will be used interchangeably. Note that this implies arc costs are strictly positive.

Let the forward star $FS(i)$ of a node $i$ be the set of outgoing arcs, that is, $FS(i) = \{b = (b_1, b_2) \in A : b_1 = i\}$, and the reverse star $RS(i)$ the set of incoming arcs: $RS(i) = \{b = (b_1, b_2) \in A : b_2 = i\}$, with the set of predecessor nodes of $i$ defined by $\Gamma^{-1}(i) = \{j \in N : (j,i) \in A\}$.

Markovian costs, also known as one-step spatial dependency, imply that the probability that an arc $b$ is in a particular state $s$ when departing at time $t$ is completely determined by the state $r$ of the previous arc traversed $a$; let this probability be denoted by $p(b, s, t, a, r)$ with $a \in RS(b_1)$, $r \in \{1, 2, \ldots, S(a, t - c_{a,t^r})\}$, $s \in \{1, 2, \ldots, S(b, t)\}$. For the sake of rigor, define a null arc $\varnothing$ as a predecessor of node $O$ which exists in one state 0 to define initial probabilities when a trip begins and there is no previous arc. Furthermore, if an arc is visited more than once, its cost is determined stochastically by **P** in all cases (that is, it is not constrained to be the same as during the first traversal). As some time necessarily passes between successive visits to an arc, this may not be entirely unrealistic.

Since we are concerned with recourse in this problem, instead of finding a single path, as in *a priori* routing problems, we seek a routing policy that contains a set of paths. Upon arrival at any node, the traveler will choose an outgoing arc to follow based on the information received thus far, in order to reach the destination in the "most reliable" manner, according to the criterion discussed in Section 3.3. The particular information the traveler will use to make a decision is an issue of importance. In networks of this form, one can find a shortest path with recourse by choosing a successor arc based only on the state of the previous

arc traveled, as done in Waller and Ziliaskopolous (2002). For finding reliable paths as we describe below, it is also necessary to allow this choice of successor arc to depend on the arrival time at a node, for reasons to be discussed shortly.

Thus, a policy prescribes a choice of outgoing arc at each node/time/predecessor combination (NTPC); that is, at each node, for each possible arrival time, and for each state of each predecessor arc of that node. Let $\Phi$ denote the set of all NTPCs implied by graph $G$, where each element is a quadruple $(i, t, a, r)$ with $i \in N$, $t \in T(i)$, $a \in RS(i)$, $r \in \{1, 2, \ldots, S(a)\}$, where $T(i)$ is the set of possible arrival times at node $i$. For any $\phi \in \Phi$, let $\phi_n$, $\phi_t$, $\phi_a$, and $\phi_r$ represent the node, time, predecessor arc, and predecessor state components of $\phi$. Further let $\mathbb{O}$ be the NTPC corresponding to the origin at departure; that is, $\mathbb{O} = (O, 0, \varnothing, 0)$, and let $\underline{\mathbb{D}}$ be the set of destination NTPCs; that is, $\underline{\mathbb{D}} = \{\phi \in \Phi: \phi_n = D\}$. Thus we can formally define a policy as a function $\pi: \Phi \rightarrow A$ such that $\pi(\phi) \in FS(\phi_n)$ for all $\phi \in \Phi$. Table A.2 shows an example of the set $\Phi$ for the small demonstration network in Appendix A.

Let $\Omega$ be the set of all possible paths from the origin to the destination (including an arbitrary number of cycles, if they exist in the graph), and of all possible states for the cost of each of these paths; that is, each element of $\Omega$ is a subset of $\Phi$ that forms a path from $\mathbb{O}$ to an element of $\underline{\mathbb{D}}$. We call each element

of $\Omega$ a *realization* of the network. For a policy $\pi$, let $p_{\omega,\pi}$ denote the probability that realization $\omega \in \Omega$ occurs when following policy $\pi$, and let $p_{\omega,\pi|\phi}$ be this probability given that NTPC $\phi$ is part of the network realization. That is, $p_{\omega,\pi}$ is the probability that the realized path from the origin to the destination corresponds with the path and arc states associated with $\omega$. Also define $X^{\omega}$ to be the cost incurred under realization $\omega$. Table A.3 shows an example of the set $\Omega$ for the small demonstration network in Appendix A.

Although $\Omega$ may be quite large, it is not directly used in the algorithm presented below; rather, the primary purpose of defining this set is to improve the clarity of the proofs of correctness that follow. Even though the size of $\Omega$ is likely to be exponential with regard to network size, this does not impede the development of an efficient algorithm.

### 3.3 The Disutility Function

In this section, we describe the development of a disutility function that penalizes what we term *deviance*, a measure of spread which is analogous to variance. In Section 3.3.1 we discuss issues that prevent direct use of variance as a disutility function. Section 3.3.2 then describes how deviance arises as a simple

modification to variance that avoids these problems, and Section 3.3.3 discusses some general properties of deviance.

### *3.3.1 Variance is Problematic*

While variance may seem to be a natural choice, a major difficulty with this approach is that Bellman's Principle does not apply: that is, subpaths of minimum variance paths may not themselves have the minimum variance. For instance, consider the network shown in Figure 3.1, where arc costs are independent. Arc C takes a cost of 10 deterministically, while arcs A and B take one of two costs with equal probability. When a traveler arrives at node 2, the cost of A is known, and a choice is made to follow either arc B or C. Thus there are four possible policies in this network; these are listed in Table 3.1, along with the associated variance, which is calculated by enumerating all possible outcomes for each policy.

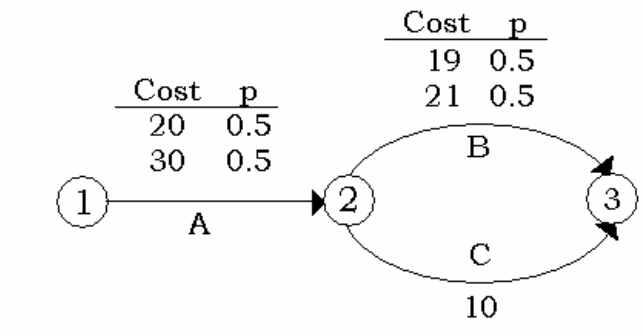


**Figure 3.1.** Bellman's Principle does not apply to variance.

**Table 3.1.** Enumerating all routing policies for the network in Figure 3.2.

| Policy | Variance |
|---|---|
| If A has cost 20, follow B; if cost is 30, follow B | 26 |
| If A has cost 20, follow B; if cost is 30, follow C | 0.5 |
| If A has cost 20, follow C; if cost is 30, follow B | 100.5 |
| If A has cost 20, follow C; if cost is 30, follow C | 25 |

Thus we see that when traveling from node 1, the minimum variance policy involves taking arc B in some cases and arc C in others. However, consider a traveler who departs from node 2. Since arc C has deterministic cost, the variance associated with choosing arc C is zero, compared to unity when choosing arc B. Thus, a traveler departing node 2 would always choose arc C, whereas one leaving node 1 might sometimes choose arc B.

Fundamentally, this occurs because variance is a measure of deviation from the expected value: by definition, the variance of a random variable $X$ is $V[X] = E[(X - E[X])^2]$. When departing from different nodes, the expected travel time to the destination is different, and therefore it is not surprising that a policy minimizing variance differs according to the departure node. Nevertheless this is undesirable from the standpoint of creating an algorithm to find a minimum variance policy. Many routing algorithms rely on Bellman's Principle to construct a path (or policy) one component at a time, and the failure of this

principle with paths of minimum variance indicates that such an approach is not promising. Furthermore, to form a policy that minimizes variance, it is necessary to know the expected cost of this policy; but given a policy and its expected cost, altering it to reduce its variance will likely change the expected cost as well. Thus, the target cost changes as we form the policy, which further confounds the problem.

### *3.3.2 Enter Deviance*

As a result, an alternative measure of reliability is sought. We propose a measure similar to variance, except that we quantify deviation from a fixed target specified *a priori*, rather than from the expected value, which is endogenously determined. For a fixed target $X^*$, define the *deviance* of a random variable $X$ as $D[X, X^*] = E[(X - X^*)^2]$. It should be noted that this metric may in fact represent a more realistic behavioral assumption, as one may not expect a traveler to be solely interested in minimizing variance of travel time without considering the expected travel time as well. The notion of choosing a desired arrival time, and minimizing deviance from that desired arrival time, implicitly models this type of behavior. Section 3.5 discusses the selection of this target, and the impact of different choices of target arrival time.

*3.3.3 General Properties of Deviance*

The primary difference between deviance and variance, and the reason that it allows the ready development of a routing algorithm, is that it measures the expected spread around an exogenous value, rather than an endogenous one. The definition of deviance is reminiscent of that of mean squared error, which is often used in parameter estimation in statistics. Indeed, the formula presented in this section for calculating deviance from the mean and variance of a distribution is identical to the formula for calculating mean squared error from the variance and bias of an estimator, although the interpretation of the quantities in the equation is somewhat different. $X^*$, which we take as a target arrival time, plays the role of the true parameter value in mean squared error, while $X$, the (random) travel time, plays the role of the estimator.

The following two results will prove useful:

---

**Proposition 3.1**. For a random variable $X$ with mean $E[X]$ and finite variance $V[X]$, the deviance from any target $X^*$ is $D\left[ X \middle| X^* \right] = V[X] + \left( E[X] - X^* \right)^2$

**Proof**. Algebraic manipulation gives

$$D\left[X, X^*\right] = E\left[\left(X - X^*\right)^2\right] = E\left[X^2\right] - 2X^* E\left[X\right] + \left(X^*\right)^2 =$$
$$= E\left[X^2\right] - \left(E\left[X\right]\right)^2 + \left(E\left[X\right]\right)^2 - 2X^* E\left[X\right] + \left(X^*\right)^2 =$$
$$= V\left[X\right] + \left(E\left[X\right]\right)^2 - 2X^* E\left[X\right] + \left(X^*\right)^2 = V\left[X\right] + \left(E\left[X\right] - X^*\right)^2$$

QED.

---

**Corollary 3.1.** For any random variable $X$ and any target $X^*$, $D[X,X^*] \geq V[X]$ with equality iff $X^* = E[X]$.

**Proof.** Follows immediately from Proposition 3.1.

---

The notion of a fixed target for travel time suggests that routing decisions should depend on arrival time at a node; for instance, if one arrives at a node at a later time, there is less time remaining before the target, and a different decision might be made. More precisely, at each node, if we arrive at time $\tau$, we attempt to minimize the deviance of the remaining travel time from $X^* - \tau$, since an amount of time $\tau$ has already elapsed since departing from the origin with the intention of traversing the network with total cost $X^*$. This dependence on arrival time does not appear in shortest path recourse problems in static networks, but is necessary when choosing minimum deviance paths for Bellman's Principle of optimality to apply.

That Bellman's Principle applies here might be intuitive. At each decision point in the network, regardless of the origin the traveler started from and the path taken, the goal is the same: to reach the destination as close to the target as possible. This is not true for variance, as the objective depends on the expected travel time, which in turn depends on the origin and all possible paths contained in the policy, and accordingly Bellman's Principle does not hold. Indeed, it is possible to rigorously show that Bellman's Principle does apply to this problem with deviance as a disutility function. Corollary 3.3 is a formal statement of this principle, and can be found in the proofs of correctness of the algorithm that is presented below.

### 3.4 Algorithm MDR-MC

With the network topology and disutility function determined, attention can now be turned to development of an algorithm to find an optimal policy. Section 3.4.1 describes the steps involved in the creation of such an algorithm. Section 3.4.2 presents pseudocode for the algorithm, and Section 3.4.3 provides proofs of correctness in acyclic graphs. Section 3.4.4 discusses complications that arise in networks that contain cycles, leading to derivation and comparison of complexity results for acyclic and cyclic networks in Section 3.4.5. A fully-

worked out example of this algorithm applied to a small network can be found in Appendix A.

### *3.4.1 Towards an Algorithm*

A modification of the standard label correcting algorithm provides a dynamic programming-like solution to the problem of finding a minimum deviance policy, leading to an algorithm we term MDR-MC (Minimum Deviance with Recourse, with Markovian Costs).

When solving all-to-one shortest path problems with a label correcting method, one begins at the destination and works backwards one node at a time to construct the shortest paths from each node. This method relies on the additivity of path costs, and, at each iteration, maintains a label at each node denoting the cost of a shortest path from that node to the destination, or ∞ if such a path has not yet been found. A scan eligible list maintains the nodes that are to be examined by the algorithm, with a shortest path chosen by considering the successor arcs to that node, and the labels at the head of those arcs. Following examination of a node, all of its predecessor arcs are added to the list.

The process we will use to decide which arc to take at a given node is as follows (see Figure 3.2 for an example which is explained below): for a given outgoing arc, although its state is uncertain, assume that we know the

distribution of path costs from the downstream node to the destination for all possible states of that arc. We can then create the distribution of path costs from the current node by merging the distributions for each downstream state, weighted according to the probability of that state, and adding the cost of that state.

In Figure 3.2, we wish to know the distribution of costs if we are at node 1, and choose to follow arc A to node 2. Note that Figure 3.2 depicts this part of the network *probabilistically*, rather than *spatially*; although there are two arcs and two nodes labeled '2', in a spatial sense these both correspond to the same arc and same node. However, they represent different states of the arc, and our assumption of spatial dependency implies that this will affect the strategy and cost distribution downstream. Once we know the distribution of costs downstream, we can calculate the deviance associated with choosing arc A, and evaluate it compared to any other available choices.

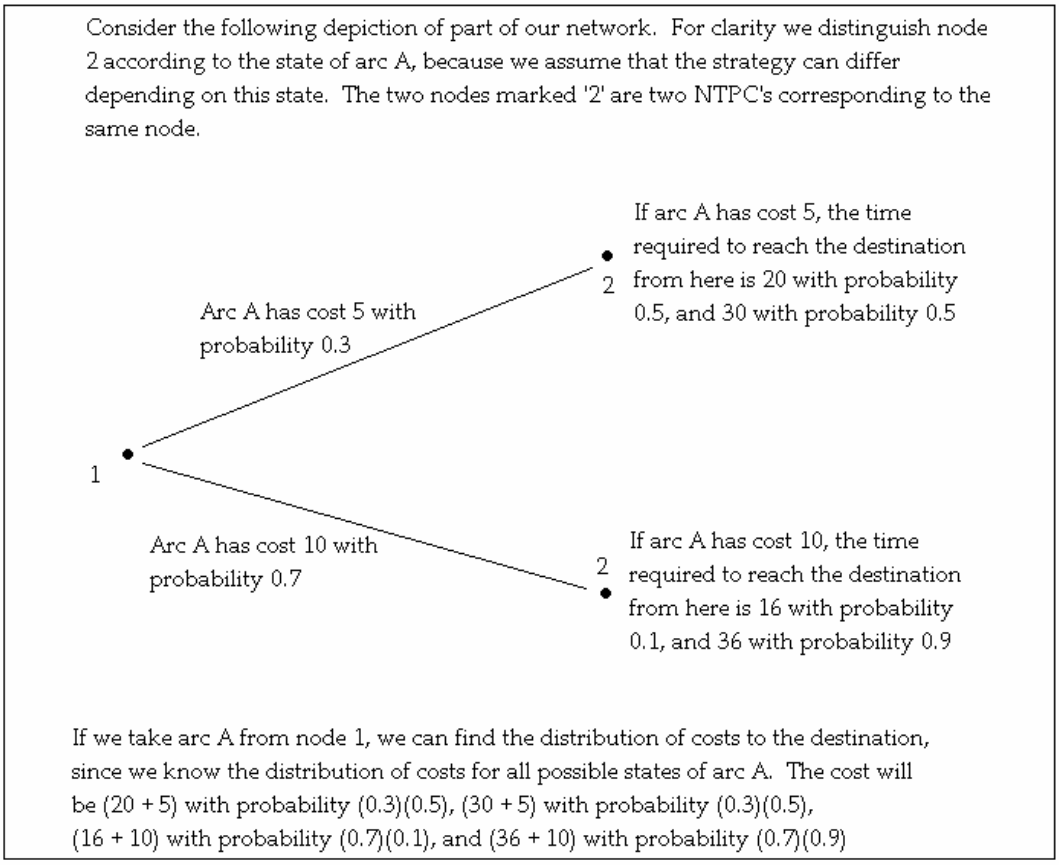


**Figure 3.2.** Determining the distribution of costs from a node.

Although the above example was simple enough that we could calculate the costs through complete enumeration, in general this becomes cumbersome as the network grows large since we must maintain the distribution of all possible costs that can occur when traveling from node 2 to the destination.

If we sought an online shortest path, as in Waller and Ziliaskopoulos (2002), rather than an online path with minimum deviance, it suffices to store the expected cost from each node to the destination as a label, because expected

value is additive. That is, even though travel cost is a random variable which takes a particular distribution, for finding shortest paths it suffices to know the expected value of this distribution at each node. This can be viewed as replacing the true distribution of costs with an equivalent distribution that places full probability on the expected value (see Figure 3.3, top).

By doing this, we avoid the need to store the complete distribution of costs at each node. For calculating the expected cost at a node, it is enough to store only the expected cost at the downstream nodes. An analogous method holds if we wish to calculate the deviance at a node, which we now describe.

From the result in Proposition 3.1, we can calculate the deviance of a random variable given its mean, variance, and a target. Thus, a natural extension of the method used to calculate shortest paths is to create a distribution with the same mean and variance as the true distribution of costs that can be described with a minimum amount of information. If the mean and variance are denoted $E[X]$ and $V[X]$, respectively, it is easily verified that a distribution that places equal probability on the two values $E[X] \pm \sqrt{V[X]}$ has the same mean, variance, and deviance (from any target) as the original distribution, while being described by only two values (see Figure 3.3, bottom).

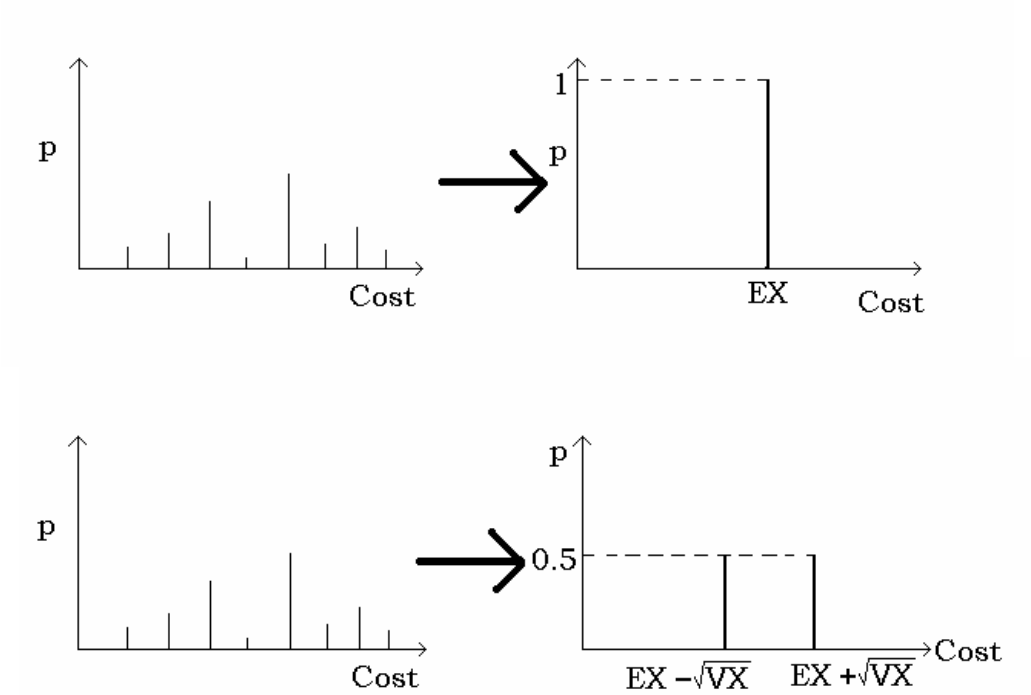


**Figure 3.3**. Two simplifying distributions which match the first and second moments.

However, we must ensure that calculating deviance of path costs using the simplified distribution is identical to using the full distribution. Lemma 3.1 and Theorem 3.1, below, accomplish this by also showing that the mean and variance calculations are also unaffected. In Theorem 3.1, one can imagine that the variables $X_1,\ldots,X_n$ represent the true distribution of costs corresponding to each of the *n* states of an arc, while $X_1^*,\ldots,X_n^*$ represent the simplified distributions which have identical mean and variance but can be described by only two values. Theorem 3.1 states that weighted sums of these random

variables also have identical mean and variance, validating the intuition that we can choose an arc by evaluating the deviance of following each of them, using the simplified distributions.

---

**Lemma 3.1.** Let $X$, $X^*$, $Y$, and $Y^*$ be independent, discrete random variables and let $Z = qX + rY$ and $Z^* = qX^* + rY^*$ for any constants $q$ and $r$. If $E[X] = E[X^*]$, $V[X] = V[X^*]$, $E[Y] = E[Y^*]$, and $V[Y] = V[Y^*]$, then:

(a) $E[Z] = E[Z^*]$

(b) $V[Z] = V[Z^*]$

**Proof.** We prove each claim in turn.

(a) $E[Z] = E[qX + rY] = q\,E[X] + r\,E[Y] = qE[X^*] + rE[Y^*] = E[qX^* + rY^*] = E[Z^*]$.

(b) From part (a) and the equation $V[A] = E[A^2] - (E[A])^2$, it is enough to show that $E[Z^2] = E[(Z^*)^2]$.

$E[Z^2] = E[(qX + rY)^2]$ (definition of expected value)

$= q^2E[X^2] + 2qrE[XY] + r^2E[Y^2]$ (linearity of expected value)

$= q^2E[X^2] + 2qrE[X]E[Y] + r^2E[Y^2]$ ($X$ and $Y$ are independent)

$= q^2E[(X^*)^2] + 2qrE[X^*]E[Y^*] + r^2E[(Y^*)^2]$

$= E[(Z^*)^2]$. This suffices to prove part (b)

QED.

---

**Theorem 3.1**. Let $X_1, X_2, \ldots, X_n$ and $X_1^*, X_2^*, \ldots, X_n^*$ be independent, discrete random variables such that $E[X_i] = E[X_i^*]$ and $V[X_i] = V[X_i^*]$ for all $i = 1, \ldots, n$.

Then if $Z = \sum_{i=1}^{n} \pi_i X_i$ and $Z^* = \sum_{i=1}^{n} \pi_i X_i^*$ for weights $\pi_i$, we have:

(a) $E[Z] = E[Z^*]$

(b) $V[Z] = V[Z^*]$

(c) $D[Z, \zeta] = D[Z^*, \zeta]$ for any target $\zeta$

**Proof**. This will be proved by induction; Lemma 3.1 establishes parts (a) and (b) of the theorem for $n = 2$; part (c) follows from Proposition 3.1. Now assume that the theorem holds for $n = k$; we will show that the case $n = k + 1$ necessarily follows. Let $A = \sum_{i=1}^{k} \pi_i X_i$ and let $B = \pi_{k+1} X_{k+1}$, with $A^*$ and $B^*$ defined similarly in terms of $X_1^*, X_2^*, \ldots, X_{k+1}^*$. Since $E[A] = E[A^*]$ and $V[A] = V[A^*]$ by the induction hypothesis and $E[B] = E[B^*]$ and $V[B] = V[B^*]$ by assumption, we have an instance of Lemma 3.1, where $Z = A + B$ and $Z^* = A^* + B^*$. Thus $E[Z] = E[Z^*]$, $V[Z] = V[Z^*]$, and, by Proposition 3.1, $D[Z, \zeta] = D[Z^*, \zeta]$ for any target $\zeta$. Thus the result holds for $n = k + 1$ as well, so by induction the theorem is true for $n \geq 2$. QED

Thus, storing two labels (representing the mean and variance of travel costs from a particular node) is sufficient for finding minimum deviance paths.

For purposes of clarity we store a third label to represent the deviance of travel cost from a node, although this is not strictly necessary since deviance can be calculated from mean and variance using Proposition 3.1. Finally, a path pointer representing the choice of outgoing arc is stored at each node, for a total of four labels. Note that the path pointer at an NTPC $\phi$ is merely the arc $\pi(\phi)$, and will be denoted as such.

Before stating and proving closed-form equations for calculating these labels, it is useful to present notation for each label. Let $E^{\pi}(i, t, a, s)$ be the label representing the expected cost to the destination from node $i$ at time $t$ when arriving via arc $a$ in state $s$ and following policy $\pi$, with $V^{\pi}(i, t, a, s)$, $D^{\pi}(i, t, a, s)$, and $\pi(i, t, a, s)$ similarly defined for variance, deviance, and the path pointer labels, where the target for deviance is not stated because it is identical throughout the network. Note that each quadruple $(i, t, a, s)$ is an NTPC, but here we write out each component for clarity. Let $E_b^{\pi}(i,t,a,s)$, $V_b^{\pi}(i,t,a,s)$, and $D_b^{\pi}(i,t,a,s)$ be the mean cost, variance, and deviance, respectively, associated with choosing arc $b$ (even if $\pi(i, t, a, s) \neq b$), and then following $\pi$ thereafter. We say that a policy $\pi$ minimizes $D$ at some $\phi \in \Phi$ if $\pi(\phi) \in \arg\min_{i \in FS(\phi_n)} D_i^{\pi}(\phi)$. For the remainder of this section, the superscript $\pi$ is suppressed for convenience.

Note that some well-defined policy for reaching the destination always exists when a node is scanned, since the the list initially consists of only the destination node, and nodes enter the scan eligible list only when one of their successor nodes is scanned.

As a corollary of Theorem 3.1, which allows us to use the simplified distributions to calculate the mean, variance, and deviance of a travel time distribution, we can find closed form expressions for these quantities in terms of the labels at a downstream node. This allows a label correcting algorithm to start at the destination, and proceed backwards through the network. At each node considered, the deviance of choosing each outgoing arc (and thereafter following the optimal policy determined thus far) can be calculated, and thus we can pick the best arc to follow, inductively extending the optimal policy one node at a time. Note that different labels will be maintained for each combination of arrival time and state of the predecessor arc; the former is necessary for reasons described above, while the latter is necessary because the probability distributions governing the cost of arc ($i$, $j$) are determined by the state of the predecessor arc.

**Corollary 3.2.** For a node $i$ being scanned at arrival time $t$, when arriving at $i$ via any predecessor arc $a$ and any state $s$, for any outgoing arc $b = (i, j)$ we have

$$E_b(i,t,a,s) = \sum_{k \in S(b)} p(b,k,t,a,s)\left[c_{b,t}^k + E\left(j, t + c_{b,t}^k, b, k\right)\right]$$

$$V_b(i,t,a,s) = \sum_{k \in S(b)} p(b,k,t,a,s)\left\{\left[E\left(j, t + c_{b,t}^k, b, k\right) + c_{b,t}^k - E_b(i,t,a,s)\right]^2 + V\left(j, t + c_{b,t}^k, b, k\right)\right\}$$

$$D_b(i,t,a,s) = V_b(i,t,a,s) + \left[E_b(i,t,a,s) - \left(X^* - t\right)\right]^2$$

**Proof.** If we choose arc $b$, for each state $k$ we can calculate the expected cost to reach the destination by adding the cost $c_b^k$ to the expected cost from $j$ at time $t + c_b^k$ when arriving via $b$ in state $k$, since path costs are additive. Thus the expression for $E_b$ follows immediately from the definition of expected value. For variance, starting from the definition and partitioning the outcomes according to the state $k$ we obtain

$$\sum_{k \in S(b)} p(b,k,t,a,s) \sum_{\omega \in \Omega} p_{\omega\left(j, t + c_{b,t}^k, b, k\right)} \left[\left(X^\omega - t\right) - E_b(i,t,a,s)\right]^2 =$$
$$\sum_{k \in S(b)} p(b,k,t,a,s)\, D\left(X^\omega \middle| E_b(i,t,a,s) \middle| \left(j, t + c_{b,t}^k, b, k\right)\right)$$

where the deviance is calculated from a target of $E_b(i,t,a,s) + t$, conditional on the event that NTPC $\left(j, t + c_b^k, b, k\right)$ is reached. However, from the above discussion, this conditional deviance is the same as that of a random variable taking values $t + c_{b,t}^k + E\left(j, t + c_{b,t}^k, b, k\right) \pm \sqrt{V\left(j, t + c_{b,t}^k, b, k\right)}$ with equal probability, so, writing the full expression for deviance, we have

$$V_b(i,t,a,s) =$$

$$\sum_{k \in S(b)} p(b,k,t,a,s) \left[ \begin{array}{l} \frac{1}{2}\left\{ t + c^k_{b,t} + E\left( j, t + c^k_{b,t}, b, k \right) + \sqrt{V\left( j, t + c^k_{b,t}, b, k \right)} - \left[ E_b(i,t,a,s) - t \right] \right\}^2 + \\ \frac{1}{2}\left\{ t + c^k_{b,t} + E\left( j, t + c^k_{b,t}, b, k \right) - \sqrt{V\left( j, t + c^k_{b,t}, b, k \right)} - \left[ E_b(i,t,a,s) - t \right] \right\}^2 \end{array} \right]$$

or, after simplification,

$$V_b(i,t,a,s) = \sum_{k \in S(b)} p(b,k,t,a,s) \left[ \left( E\left( j, t + c^k_{b,t}, b, k \right) + c^k_{b,t} - E_b(i,t,a,s) \right)^2 + V\left( j, t + c^k_{b,t}, b, k \right) \right]$$

which is as stated above. Finally, the expression for deviance follows immediately from Proposition 3.1.

---

### *3.4.2 Pseudocode for MDR-MC*

With the above theorem in mind, the necessary modifications made to the label correcting algorithm can be described. When examining an NTPC, we consider all possible outgoing arcs, and read the labels for the downstream node, learning for each the expected cost to the destination and the variance in this cost. From this we can calculate the expected cost, variance, and deviance from leaving the node currently under consideration via that arc using the formulas from Corollary 3.2. After doing this for all emanating arcs, we find one with

minimum $D$, and update the path pointer, labels, and scan eligible list accordingly.

Notice that the scan eligible list consists of node-time pairs $(i, t)$ with $t \in T(i)$; when a node-time pair is scanned, the policy for all NTPCs corresponding to $(i, t)$ is updated. That is, a policy for node $i$ at time $t$ is determined for all states of all incoming arcs. This transformation to a time-expanded network improves the clarity of the proofs of correctness and complexity.

Pseudocode for the algorithm is presented in Figure 3.4; a fully worked-out example of the application of this algorithm to a small network is found in Appendix A.

### *3.4.3 Correctness in Acyclic Networks*

Since travel times are strictly positive, the time-expanded graph is always acyclic; however, complications arise when there are cycles in the underlying network, since the time-expanded graph becomes infinite. (As the time-expanded graph is necessarily acyclic, from here on "cyclic" and "acyclic" describe the topology of the underlying network, unless otherwise stated). The following proofs are given for acyclic networks; the next section describes the case of networks with cycles.

Initialization.

$\forall$ $a,s,t$ with $a \in RS(D)$, $s \in S(a)$, $t \in T(D)$, let $E(D,t,a,s) := 0$, $V(D,t,a,s) := 0$,
$D(D,t,a,s) := (t - X^*)^2$

$\forall$ $i,t,a,s$ with $i \in (N \setminus D)$, $t \in T(j)$, $a \in RS(i)$, $s \in S(a)$, let $E(i,t,a,s) := \infty$, $V(i,t,a,s) := \infty$,
$D(i,t,a,s) := \infty$.

$SEL := \{(i,t): i \in \Gamma^{-1}(D), t \in T(i)\}$

Iteration:

while $SEL \neq \emptyset$

Select $(i,t) \in SEL$ and set $SEL := SEL \setminus (i,t)$

For $\forall$ $a,b,s$ with $a \in RS(i)$, $b \in FS(i)$, $s \in S(a,t)$

$$E_b(i,t,a,s) := \sum_{k \in S(b)} p(b,k,t,a,s)\left(c^k_{b,t} + E\left(j, t + c^k_{b,t}, b, k\right)\right)$$

$$V_b(i,t,a,s) := \sum_{k \in S(b)} p(b,k,t,a,s)\left(\left(E\left(j, t + c^k_{b,t}, b, k\right) + c^k_{b,t} - E_b(i,t,a,s)\right)^2 + V\left(j, t + c^k_{b,t}, b, k\right)\right)$$

$$D_b(i,t,a,s) := V_b(i,t,a,s) + \left[E_b(i,t,a,s) - \left(X^* - t\right)\right]^2$$

if $D_b(i,t,a,s) < D(i,t,a,s)$

$E(i,t,a,s) := E_b(i,t,a,s)$

$V(i,t,a,s) := V_b(i,t,a,s)$

$D(i,t,a,s) := D_b(i,t,a,s)$

$\pi(i,t,a,s) := b$

$SEL := SEL \cup \{(h,u): h \in \Gamma^{-1}(i), u = t - c^r_{(h,i)}$ for some $r \in \{1, 2, \ldots, S((h,i), t - c^r_{(h,i)})\}$

**Figure 3.4**. Pseudocode for Algorithm MDR-MC

We prove the correctness of the algorithm for acyclic networks in several steps: we first show that it must terminate, then, through four lemmas, proceed to show that the policy it generates is optimal in the sense that no other policy has a lower deviance – that is, no other policy has a smaller $D$ label at $\mathbb{O}$.

**Theorem 3.2**. In acyclic networks, algorithm MDR-MC terminates after finitely many iterations.

**Proof**. By contradiction, assume that the algorithm never terminates. This implies that *SEL* is never empty. Since an element is removed from *SEL* at each iteration, if it is never empty there must be infinitely many additions to *SEL*. Since the number of node-time pairs in such networks is finite, some node-time pair must be added to *SEL* infinitely many times; however, this contradicts the assumption that the network is acyclic, since the number of times a node-time pair is scanned is bounded by the maximum number of arcs that can be traveled before the destination is reached, which is finite. QED

We now prove that the algorithm generates an optimal policy in three steps: first, we establish that upon termination, the algorithm will produce a policy that minimizes $D$ at each $\phi \in \Phi$; next, we show that all policies that minimize $D$ at each NTPC have the same deviance; that is, ties can be broken arbitrarily. Finally, we show that any policy which does not minimize $D$ at each NTPC is suboptimal. Together, this establishes that the output of the algorithm is optimal.

Before presenting this proof we define the following notation: let $\Phi_a^{+1}(\phi)$ be the set of NTPCs that can be immediately reached by departing NTPC $\phi$ via arc *a*, that is, the downstream node of *a* with times corresponding to all possible states of arc *a*. For notational convenience define $\rho_{\phi,S,a|\phi_0}$ to be the probability of reaching NTPC $\phi$ immediately following $\phi_0$ by choosing arc *a*, corresponding to the probabilities $p(a, s, (\phi_0)_t, (\phi_0)_a, (\phi_0)_r)$ with *s* such that $\phi_t = (\phi_0)_t + c_a^s$.

---

**Lemma 3.2**. Upon termination of the algorithm, we have a policy $\pi$ that minimizes *D* at each $\phi \in \Phi$.

**Proof**. By contradiction, assume that the algorithm has terminated, but there exists $\phi \in \Phi$ such that the resulting policy does *not* minimize *D* there; that is, there exist arcs *b*, *c* in $FS(\phi_n)$ such that $\pi(\phi) = b$ and $D_b^\pi(\phi) > D_b^\pi(\phi)$. Let *k* be the node reached by following arc *c*. Since a path pointer exists for $(\phi_n, \phi_t)$ it must have been in *SEL* at least once; consider the last time that this pair was chosen from *SEL*. Since successor node *b* was instead of *c*, we know $D_b^\pi(\phi) \le D_b^\pi(\phi)$ when *i* was last scanned. However, at termination, by assumption we have $D_b^\pi(\phi) > D_b^\pi(\phi)$, indicating that $D_b^\pi(\phi)$ must have changed again before termination, which implies that some *D* label at a node-time pair $(k, \phi_t + c_{c,\phi_t}^r)$ has changed for some $r \in \{1, 2, \ldots, S(c, \phi_t)\}$. But if this happened, $(\phi_n, \phi_t)$ would

reenter SEL, contradicting the assumption that this was the last time $(\phi_n, \phi_t)$ was chosen from *SEL*. Thus, when the algorithm terminates, the resulting policy minimizes *D* for each $\phi$ in $\Phi$.

---

**Lemma 3.3**. For any $\phi_0 \in \Phi$ and arc $b \in FS(\phi_0)_n$, and any two policies *A* and *B* for which $D^A(\phi) = D^B(\phi)$ for all $\phi \in \Phi_b^{+1}(\phi_0)$, we have $D_b^A(\phi_0) = D_b^B(\phi_0)$.

**Proof**. The deviance label at any NTPC $\psi$ under a policy $\pi$ can be expressed as $\sum_{\omega \in \Omega} p_{\omega,\pi|\psi} \left( X_\omega - X^* \right)^2$. Therefore, we have

$$D_b^A(\phi_0) = \sum_{\omega} p_{\omega,A|\phi_0} \left( X_\omega - X^* \right)^2 = \sum_{\omega} \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} p_{\omega,A|\phi} \left( X_\omega - X^* \right)^2 =$$

$$= \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} \sum_{\omega} p_{\omega,A|\phi} \left( X_\omega - X^* \right)^2 = \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} \sum_{\omega} p_{\omega,B|\phi} \left( X_\omega - X^* \right)^2 = D_b^B(\phi_0)$$

since $\sum_{\omega} p_{\omega,A|\phi} \left( X_\omega - X^* \right)^2 = \sum_{\omega} p_{\omega,B|\phi} \left( X_\omega - X^* \right)^2$ for all $\phi \in \Phi_b^{+1}(\phi_0)$ by assumption.

---

**Lemma 3.4**. All policies that minimize $D(\phi)$ at all $\phi \in \Phi$ have the same $D(\psi)$ for all $\psi \in \Phi$, and, in particular, the same $D(\mathbb{O})$.

**Proof**. Let *A* and *B* be two distinct policies that minimize *D* at each NTPC. (The claim is trivial if the set of such policies is a singleton). Let $\Phi_K$ denote the set of NTPCs that are at most *K* arcs away from the destination. Since the network is acyclic, each NTPC belongs to some $\Phi_K$. We now show that $D^A(\phi) = D^B(\phi)$ for all

$\phi \in \Phi$ by strong induction. Consider any $\phi_1 \in \Phi_1$. Since $\Phi^{+1}(\phi_1) \subseteq \underline{\mathbb{D}}$ and clearly $D^A(\phi) = D^B(\phi)$ for all $\phi \in \underline{\mathbb{D}}$, $D^A(\phi) = D^B(\phi)$) by Lemma 3.3. Now assume that $D^A(\phi) = D^B(\phi)$ for all $\phi \in \bigcup_{k=1}^{I} \Phi_k$. Consider any $\phi_{I+1} \in \Phi_{I+1}$. By the induction hypothesis $D^A(\phi) = D^B(\phi)$ for $\phi \in \Phi^{+1}(\phi_{I+1}) \subseteq \bigcup_{k=1}^{I} \Phi_k$, so by Lemma 3.3 we also have $D^A(\phi) = D^B(\phi)$. Thus, for all $K$, $D^A(\phi) = D^B(\phi)$ for $\phi \in \Phi_K$. Since each $\phi \in \Phi$ belongs to some $\Phi_K$, this proves the claim.

---

**Lemma 3.5**. Any policy that does not minimize $D(\phi)$ for all $\phi \in \Phi$ reached with positive probability is suboptimal.

**Proof**. Let $A$ be a policy as described above. Then there exist some $\phi^* = (i, t, a, s)$ and arcs $b$, $c$ such that $A(\phi) = b$ but $D_b^A(\phi) > D_c^A(\phi)$. Define a new policy $B$ that is identical to $A$ except $B(\phi) = c$. We now show that $D^B(\mathbb{O}) < D^A(\mathbb{O})$, thereby proving the lemma.

Recall that $\Omega$ is the set of all realizations of the network. Partition $\Omega$ into sets $\Pi_1$ and $\Pi_2$ where $\Pi_1$ contains all realizations containing $\phi^*$, and $\Pi_2 = \Omega \setminus \Pi_1$. Thus, we have

$$D^A(\mathbb{O}) = \sum_{\omega \in \Pi_1} p_{\omega,A} \left( X_\omega - X^* \right)^2 + \sum_{\omega \in \Pi_2} p_{\omega,A} \left( X_\omega - X^* \right)^2$$

$$D^B(\mathbb{O}) = \sum_{\omega\in\Pi_1} p_{\omega,B}\left(X_\omega - X^*\right)^2 + \sum_{\omega\in\Pi_2} p_{\omega,B}\left(X_\omega - X^*\right)^2$$

Since $A$ and $B$ are identical except at $\phi^*$, $p_{\omega,A} = p_{\omega,B}$ for all $\omega \in \Pi_2$, so

$$D^A(\mathbb{O}) - D^B(\mathbb{O}) = \sum_{\omega\in\Pi_1} p_{\omega,A}\left(X_\omega - X^*\right)^2 - \sum_{\omega\in\Pi_1} p_{\omega,B}\left(X_\omega - X^*\right)^2 =$$

$$\sum_{\omega\in\Pi_1} p_{\omega,A}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2 - \sum_{\omega\in\Pi_1} p_{\omega,B}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2.$$

Since $A$ and $B$ differ only at $\phi^*$, we have $\sum_{\omega\in\Pi_1} p_{\omega,A} = \sum_{\omega\in\Pi_1} p_{\omega,B}$, and by assumption both sums are strictly positive. Thus we divide the first term by $\sum_{\omega\in\Pi_1} p_{\omega,A}$ and the second by $\sum_{\omega\in\Pi_1} p_{\omega,B}$, preserving the sign, which yields

$$\frac{\sum_{\omega\in\Pi_1} p_{\omega,A}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2}{\sum_{\omega\in\Pi_1} p_{\omega,A}} - \frac{\sum_{\omega\in\Pi_1} p_{\omega,B}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2}{\sum_{\omega\in\Pi_1} p_{\omega,B}} =$$

$$= \sum_{\omega\in\Pi_1} p_{\omega,A|\phi^*}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2 - \sum_{\omega\in\Pi_1} p_{\omega,B|\phi^*}\left(\left(X_\omega - T^*\right) - \left(X^* - T^*\right)\right)^2 =$$

$$= D_b^A(\phi^*) - D_c^A(\phi^*)$$

since $A$ and $B$ differ only at $\phi^*$. But $D_b^A(\phi^*) > D_c^A(\phi^*)$ so $D^A(\mathbb{O}) - D^B(\mathbb{O}) > 0$. Thus $D^A(\mathbb{O}) > D^B(\mathbb{O})$, completing the proof that $A$ is suboptimal.

**Theorem 3.3**. Any policy produced by algorithm MDR-MC is optimal.

**Proof**. Let the set of all possible policies be partitioned into sets *M* and *N*, where *M* consists of all policies that minimize *D* at all NTPCs, and *N* consists of all other policies. Since there are only finitely many policies, there must be some policy whose deviance is no greater than that of any other policy. By Lemma 3.5, no policy in set *N* is optimal; thus, some policy in *M* is optimal. By Lemma 3.4, all policies in *M* have the same deviance; hence all policies in *M* are optimal. By Lemma 3.2, the algorithm will produce a policy in *M*; thus it will return an optimal policy. QED

From the above results, the extension of Bellman's Principle follows as an easy corollary. Given an NTPC $\phi$, let $\Phi^+(\phi,\sigma)$ be the set of all NTPCs that are reached from $\phi$ with positive probability when following policy $\pi$.

**Corollary 3.3 (Bellman's Principle)**. If *A* is an optimal policy for reaching the destination from origin $\mathbb{O}_1 = \phi_1 \in \Phi$, then for any $\phi_2 \in \Phi^+(\phi_1, A)$, the policy *B* with $B(\phi) = A(\phi)$ for all $\phi \in \Phi^+(\phi_2,A)$ is optimal for reaching the destination from origin $\mathbb{O}_2 = \phi_2$.

**Proof**. Assume not. Then there exists an optimal policy $C$ such that $D^C(\phi_2) < D^B(\phi_2)$. From Lemma 4, for all $\phi \in \Phi^+(\phi_2, A)$, $C(\phi) \in \arg\min_{i \in FS(\phi)} D_i(\phi)$. Since $B$ is suboptimal, there exists some $\phi^* \in \Phi^+(\phi_2, A)$ for which $B(\phi) \neq C(\phi)$, and furthermore, by Lemma 3, for which $B(\phi^*) \notin \arg\min_{i \in FS(\phi)} D_i(\phi^*)$. However since $A(\phi) = B(\phi)$ for all $\phi \in \Phi^+(\phi_2, A)$ this implies $A(\phi^*) \notin \arg\min_{i \in FS(\phi)} D_i(\phi^*)$, so, by Lemma 3.5, $A$ is suboptimal, contradicting the original assumption. Thus no such policy $C$ exists, so $B$ is itself optimal.

---

*3.4.4 Complications in Cyclic Networks*

The presence of cycles can cause difficulties in online algorithms: for instance, in online shortest path problems, in general one cannot rule out paths that include cycles. For instance, the example in Figure 3.5, based on Waller and Ziliaskopoulos (2002), has an optimal policy where a cycle is repeated arbitrarily many times. In this graph, after observing the state of arc (2, 3) the state of (3, 4) is known. When arriving at node 3, if arc (2, 3) had cost 1, the traveler knows that the cost of arc (3, 4) is very high; thus, the optimal strategy is to travel back to node 1, hoping that after another traversal, the state of (3, 4) will be lower. (Note the application of the reset assumption here: the costs of arcs (2, 3) and

(3, 4) are not constrained to be the same if visited more than once.) However, at the second traversal, there is still no guarantee that (3, 4) will have low cost; and the cycle could be traversed a third time, and so on. Thus, there is no upper bound on the maximum number of cycles that will be present in the optimal solution, although the probability of a large number of traversals of a cycle is small.

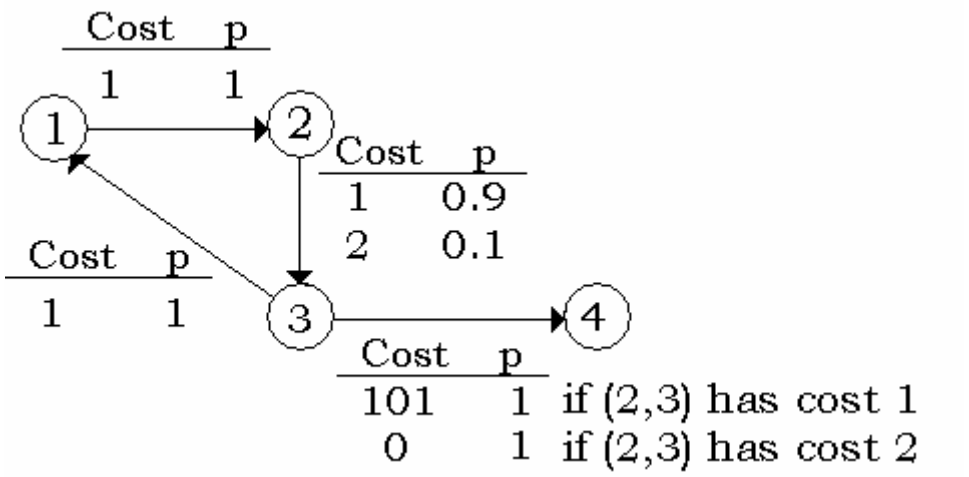


**Figure 3.5**. Infinite cycling in an online shortest path.

Working in an expanded network where each ($n$, $t$) pair is a separate vertex guarantees an acyclic graph because arc costs are strictly positive; nevertheless, if cycles exist in the original network, there will exist nodes whose set of possible arrival times is infinite. In particular, the set $\mathbb{D}$ will include infinitely many NTPCs, and thus the above algorithm will not terminate.

However, since the goal is to minimize deviance from a fixed target $X^*$, bounds can be formed on the probability that a traveler will remain on the network after a certain time $T$. This result can in turn be used to bound the suboptimality induced by forcing trips to end before time $T$. Therefore, in cyclic networks, one can constrain all trips to end before some time, apply the algorithm to the constrained graph (which is finite), and obtain a policy whose deviance lies within a specified tolerance of the optimum. Derivation of these bounds follows a more detailed description of the constrained problem.

Given an infinite time-expanded graph, we create a *T-constrained graph* as follows: we first delete all NTPCs whose time component is greater than $T$, as well as all arcs that terminate (with positive probability) at any of the deleted NTPCs. If, as a result of this step, any node no longer has any outgoing arcs, it is itself deleted, along with any arcs who have this node as a head. This step is repeated as often as is necessary to ensure that all non-destination nodes have at least one outgoing arc. Since the above procedure only preserves NTPCs with time component no greater than $T$, the resulting graph will be finite, and the above algorithm is certain to terminate when applied to the $T$-constrained graph. We define a *T-feasible policy* to be a policy on a $T$-constrained graph. Since $T$-feasible policies are also feasible for the unconstrained graph, the optimal

*T*-feasible policies must have a deviance greater than the optimal unconstrained policy. We seek a bound on this difference.

---

**Lemma 3.6**. For $\tau > X^* + Mn$, the probability that the traveler will complete a trip after time $\tau$ when following an optimal policy $\pi^*$ is no greater than $\frac{(Mn)^2}{(\tau - X^*)^2}$.

**Proof.** Consider the following $\tau$-feasible policy: travel deterministically from the origin to any node that is part of a cycle, then traverse this cycle repeatedly until the time is greater than $X^*$, at which point any acyclic path is followed to the destination. Since the first part of this trip has cost no greater than $X^* + M$, and the second part has cost no greater than $M(n-1)$, this policy will result in arrival at the destination between times $X^*$ and $X^* + Mn$ and, thus, will have deviance no greater than $(Mn)^2$. Therefore, the optimal policy can certainly have deviance no greater than $(Mn)^2$. Therefore we have

$$\sum_{\omega} p_{\omega,\pi^*}\left(X^{\omega} - X^*\right) = \sum_{\omega: X^{\omega} \le \tau} p_{\omega,\pi^*}\left(X^{\omega} - X^*\right) + \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*}\left(X^{\omega} - X^*\right) \le (Mn)^2$$

$$\Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*}\left(X^{\omega} - X^*\right)^2 \le (Mn)^2 \Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*}\left(\tau - X^*\right)^2 \le (Mn)^2 \Rightarrow$$

$$\Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*} \le \frac{(Mn)^2}{(\tau - X^*)^2}$$

proving the lemma.

---

Next we consider a bound on the difference between the deviance of an optimal $T$-feasible policy $(\pi^*)^T$ and an optimal unconstrained policy $\pi^*$ when $T > X^* + Mn$. To do this, we construct a (possibly) suboptimal $T$-feasible policy $\pi^T$ and bound the difference between $V^*(\pi^*)$ and $V^*(\pi^T)$. Let $\pi^T(\phi) = \pi^*(\phi)$ for $\phi$ such that $(\phi)_t \le T - Mn$; when $(\phi)_t > T - Mn$ we have the traveler deterministically follow any acyclic path to the destination; that is, the optimal unconstrained policy is used until time $T - Mn$, after which a deterministic path is followed. Such a policy is $T$-feasible since one can always reach the destination from any node incurring cost less than $Mn$.

**Theorem 3.4**. The difference between the deviance of an optimal policy and an optimal $T$-feasible policy is no greater than $\dfrac{(Mn)^4 + 2(Mn)^3(T - X^*)}{(T - Mn - X^*)^2}$.

**Proof.** Partition $\Omega$ into two sets $\Omega_1$ and $\Omega_2$ with $\Omega_1 = \{\omega: X^\omega \ge T - Mn\}$ and $\Omega_2 = \{\omega: X^\omega < T - Mn\}$. (Note that $\Omega$ is defined for the unconstrained graph; but this state space will also be used for the $T$-feasible policy). Thus

$$D(\pi^*) = \sum_{\omega \in \Omega_1} p_{\omega,\pi^*} \left(X^\omega - X^*\right)^2 + \sum_{\omega \in \Omega_2} p_{\omega,\pi^*} \left(X^\omega - X^*\right)^2$$

and

$$D\left(\pi^{T}\right) \le \left(T - X^{*}\right)^{2} \sum_{\omega \in \Omega_1} p_{\omega,\pi^{*}} + \sum_{\omega \in \Omega_2} p_{\omega,\pi^{*}} \left(X^{\omega} - X^{*}\right)^{2}$$

since $p_{\omega,\pi^{*}} = p_{\omega,\pi^{T}}$ for $\omega \in \Omega_2$. Therefore

$$D(\pi^{T*}) - D(\pi^{*}) \le D(\pi^{T}) - D(\pi^{*}) \le \sum_{\omega \in \Omega_1} p_{\omega,\pi^{*}} \left[ \left(T - X^{*}\right)^{2} - \left(X^{\omega} - X^{*}\right)^{2} \right] \le$$

$$\le \sum_{\omega X^{\omega} \in [T-Mn,T]} p_{\omega,\pi^{*}} \left[ \left(T - X^{*}\right)^{2} - \left(X^{\omega} - X^{*}\right)^{2} \right]$$

the last inequality because $\left(T - X^{*}\right)^{2} - \left(X^{\omega} - X^{*}\right)^{2}$ is negative when $X^{\omega} > T$.

Therefore we have

$$D(\pi^{T*}) - D(\pi^{*}) \le \sum_{\omega X^{\omega} \in [T-Mn,T]} p_{\omega,\pi^{*}} \left[ \left(T - X^{*}\right)^{2} - \left(X^{\omega} - X^{*}\right)^{2} \right] \le$$

$$\left[ \left(T - X^{*}\right)^{2} - \left(T - Mn - X^{*}\right)^{2} \right] \sum_{\omega X^{\omega} \in [T-Mn,T]} p_{\omega,\pi^{*}} \le \left[ (Mn)^{2} + 2Mn\left(T - X^{*}\right) \right] \frac{(Mn)^{2}}{\left(T - Mn - X^{*}\right)^{2}}$$

using Lemma 3.6, or $D(\pi^{T*}) - D(\pi^{*}) \le \dfrac{(Mn)^{4} + 2(Mn)^{3}\left(T - X^{*}\right)}{\left(T - Mn - X^{*}\right)^{2}}$ QED

---

Note that the bound in Theorem 3.4 becomes arbitrarily small for sufficiently large $T$. Thus for a given tolerance $\varepsilon$, we can find the $T$-constrained graph whose optimal policy is within $\varepsilon$ of the optimal unconstrained policy using the formula in the corollary below:

**Corollary 3.4**. For $\varepsilon > 0$, $D(\pi^T*) - D(\pi^*) \leq \varepsilon$ if

$$T > X^* + Mn + \frac{(Mn)^2}{\varepsilon}\left(Mn + \sqrt{(Mn)^2 + 3\varepsilon}\right)$$

**Proof**. Setting the bound in Theorem 3.4 to $\varepsilon$ and solving for $T$ yields the result.

The bounds developed in this section are quite loose for typical networks that are encountered in practice, and one can almost certainly obtain tighter bounds by exploiting the topology of a particular network. Nevertheless, the existence of a bound for general networks indicates that a solution can be found within any desired tolerance in finite time.

*3.4.5 Complexity Results*

For the proofs of complexity, let $T = \max_{\phi \in \Phi}\{\phi_t\}$ and $T = \max_{a \in A, t \in T(a)}\left\{\left|S(a,t)\right|\right\}$ refer to the latest arrival time at any node and the maximum number of states an arc exists in, respectively. In cyclic graphs assume we are solving a $T$-constrained problem. Recall that $M$ is the largest possible cost of any arc. The following proposition demonstrates that MDR-MC has pseudopolynomial complexity in both acyclic and cyclic graphs when the tolerance $\varepsilon$ is fixed.

**Proposition 3.2**. Algorithm MDR-MC has complexity $O(nmTS^2\min\{n,T\})$ when a FIFO SEL is used.

**Proof**. This proof of complexity mirrors that of the label-correcting deterministic shortest path algorithm presented in Ahuja et al (1993). With a FIFO SEL, at most $\min\{n, T\}$ passes are required; at worst, each pass will require a summation to be made for each arrival time at the head of each arc, for each predecessor of this arc, and for each state this predecessor. With $m$ arcs, at most $T$ arrival times for each node, at most $n$ predecessor arcs, and at most $S$ states for any arc, each pass requires at most $nmTS$ summations, each of which involves at most $S$ terms. Hence each pass requires at most $nmTS^2$ calculations, for a total complexity of $O(nmTS^2\min\{n,T\})$

---

**Proposition 3.3**. For acyclic graphs, MDR-MC has complexity $O(Mn^3mS^2)$.

**Proof**. In acyclic networks, $T \le Mn$, leading to the stated complexity.

---

**Proposition 3.4**. In networks with cycles, for a tolerance $\varepsilon > 0$, algorithm MDR-MC determines a policy within $\varepsilon$ of the optimal policy with complexity $O(M^3n^5mS^2/\varepsilon)$ as $\varepsilon \to 0$.

**Proof**. Apply MDR-MC to a $T$-constrained graph with $T$ being the smallest integer satisfying the bound in Corollary 3.4. The result follows immediately from Proposition 3.2.

---

### 3.5 Target Selection

Although the choice of target arrival time $X^*$ does not impact the worst-case running time of the algorithm, it certainly has substantial influence on the solution characteristics. While the particular behavior of a policy for varying $X^*$ is highly network-dependent, some generalizations can be drawn. For instance, if $X^*$ is less than the quickest possible path through the network, any deviance in travel time is due to exceeding the target; thus, we expect that the solution will resemble a policy that minimizes expected cost, although since we are squaring the penalty travel times the optimal solutions need not be identical. Likewise, in cyclic networks, if $X^*$ is greater than the longest acyclic path through the network, it is likely that cyclic paths will be chosen most of the time.

In some applications, the appropriate target arrival time may correspond to a clear behavioral objective, as in the case of commuters attempting to travel from home to work on time, or supply chains involving goods that are perishable or involve high storage costs. In general a natural choice for $X^*$ is the expected cost of an online shortest path (OSP) policy.

Using the expected cost of the OSP solution as the target carries an additional benefit, namely, that the policy obtained will have lower variance than the OSP solution, as is shown below:

**Proposition 3.5**. If the minimum deviance with recourse (MDR) policy is found with the expected OSP cost as the target, its variance will be less than that of the online shortest path policy.

**Proof**. From Corollary 3.1, the variance of the MDR policy is a lower bound for its deviance from the expected cost of the OSP policy; further, since the MDR policy minimizes the deviance from this target, its deviance is less than that of the OSP policy from the same target. But this target is simply the expected OSP cost, so, by Corollary 3.1, this deviance is the same as its variance. Thus the variance of the MDR policy is no greater than the variance of the OSP policy.

By varying the target, other policies can be obtained. For instance, choosing a target of zero essentially transforms the problem to finding a risk-averse online shortest hyperpath, where the penalty function is the square of the travel time. One would expect a slight increase in the expected travel time as compared to the OSP solution, in exchange for a reduction in the frequency of very long trips.

### 3.6 One-Sided Deviance

One possible criticism of deviance is that it penalizes early and late arrivals equally; however, here we show that a simple modifications of the network allows one to penalize only late arrivals; that is, to minimize $E((X - X^*)^+)^2$ rather than $E(X - X^*)^2$, where $(X - X^*)^+ = \max\{X - X^*, 0\}$. To accommodate this, one may add an arc from the destination node to itself which always has unit cost, which allows indefinite waiting at this node. Effectively, this eliminates any penalty for early arrival at the destination, as one can traverse the cycle repeatedly until the target is attained.

This does not significantly affect the performance of the algorithm; although the network becomes cyclic, this cycle will be traversed if and only if one arrives at the destination before the target. Thus, in networks that were originally acyclic, no change needs to be made as long as $T > X^*$ (otherwise one may add the node-time pairs $(D, \tau)$ to the network for $\tau \in \{T + 1, \ldots, X^*\}$); in cyclic networks, the $T$ value chosen to meet the desired tolerance does not change. Although the computation time increases slightly, as the node-time pairs $(D, t)$ must enter *SEL* and be examined in turn, the worst-case complexity is not affected.

## 3.7 Conclusion

In this chapter we have presented a pseudopolynomial algorithm, MDR-MC, which finds an optimal routing policy in a stochastic, time-varying network where arc costs are Markovian and the squared difference between actual and desired arrival time forms the disutility function. Correctness and complexity results were proved, and procedures to allow cycles in the network were created as well.

Although variance formed the basis for deviance, the quantity actually minimized by this algorithm, it was shown that by choosing the target arrival time to be the expected cost of an online shortest path solution, the variance of the policy found by MDR-MC is necessarily lower than the variance of the online shortest path solution. A simple extension to allow a penalty only for late arrivals was also discussed, as in some applications early arrivals may not be particularly burdensome.

In the next chapter, we generalize the procedure and algorithm to account for a broader class of utility functions, namely, arbitrary piecewise polynomials.

# Chapter 4: A General Algorithm: MPPR-MC

## 4.1 Introduction

The previous chapter contained the development of an algorithm to find a policy with minimum deviance, an important special case of a polynomial utility function which also served to illustrate some of the issues involved in finding an optimal routing policy. Here we formulate a more general model, although many of the proofs are similar. The notation and network topology are identical to that presented in Chapter 3.

Section 4.2 rigorously describes the type of disutility function we consider here, while Section 4.3 investigates the applicability of Bellman's Principle with such a disutility function, and provides the means to calculate the expected disutility of a path, analogous to Corollary 3.2, for disutility functions of one piece. Accommodating piecewise polynomial functions is described in Section 4.4, and Section 4.5 presents the algorithm itself, along with proofs of correctness and complexity, and discussion of issues present in cyclic networks. Section 4.6 produces results allowing us to bound the error induced by approximating nonpolynomial functions with a truncated Taylor polynomial. Finally, Section 4.7 summarizes the findings described in this chapter.

## 4.2 Piecewise Polynomial Disutility Functions

In Chapter 3, the squared difference between the actual and desired arrival time was penalized; this can be expressed as the disutility function $D(t) = (t - t^*)^2 = t^2 - 2t(t^*) + (t^*)^2$, where $t$ is the actual arrival time, and $t^*$ is the target. Algorithm MDR-MC then found a policy which minimized $E[D(t)]$.

In this chapter, we consider a broader class of disutility functions, the piecewise polynomial functions. We take the common definition of a real polynomial $P$ of degree $d$ ($P(x) = \sum_{i=0}^{d} a_i x^i$ for real constants $a_0, \ldots, a_d$ with $a_d \neq 0$.) A piecewise polynomial function is one comprised of a finite number of polynomials defined over intervals in the real line. Formally, a piecewise polynomial function of $N$ pieces, each with degree $d_j$ on disjoint intervals $A_j$ with positive measure, $j \in \{1, \ldots, n\}$, can be expressed as $P(x) = \sum_{j=1}^{N} \sum_{i=0}^{d_j} a_{ji} x^i I_{x \in A_j}(x)$ where $I_{x \in A_j}(x)$ is an indicator function equal to unity if $x \in A_j$ and zero otherwise. Clearly, deviance is a special case of this, with $N = 1$ piece of degree $d_1 = 2$ defined on interval $A_1 = [0, \infty)$ and coefficients $a_{11} = 1$, $a_{12} = -2t^*$, and $a_{13} = (t^*)^2$.

For this algorithm, we will require that the polynomial be defined for all possible arrival times in the network.

### 4.3 Bellman's Principle

The first step in developing a label-correcting algorithm for this case is to determine how to extend existing paths from intermediate nodes to the destination, allowing a solution via backward induction. For the case of deviance, the equations in Corollary 3.2 sufficed to calculate the deviance of adding the current arc to an existing path. In this section we discuss a method for doing this with polynomials; for simplicity, in this section we only consider polynomials of one piece. Extension to piecewise polynomials is accomplished in Section 4.4.

The equations in Corollary 3.2 were obtained by forming a "simplifying distribution" which had the same mean and variance as the true distribution of path costs, but which only required storage of two labels, obviating the need to enumerate all possible costs and vastly reducing the storage and computational requirements. This sufficed because evaluation of the expected value of the disutility function only required knowledge of the first two moments of the true distribution of costs:

$$E\left[D(X)\right] = E\left[X^2 - 2XX^* + \left(X^*\right)^2\right] = E\left[X^2\right] - 2X^*E\left[X\right] + \left(X^*\right)^2$$

In general, knowing the first $n$ moments of the distribution is enough to evaluate the expected value of a polynomial disutility function of degree $n$:

$$E\left[P(x)\right] = E\left[\sum_{i=0}^{d} a_i x^i\right] = \sum_{i=0}^{d} a_i E\left[x^i\right]$$

With deviance the calculation was facilitated through the use of a simplifying distribution whose first two moments were identical with that of the true distribution.

However, directly extending this technique is not straightforward. The support points of the simplifying distribution were determined by simultaneous solution of the equations

$$\begin{cases} \frac{1}{2}\left(x_1 + x_2\right) = E\left[X\right] \\ \frac{1}{2}\left(x_1^2 + x_2^2\right) = E\left[X^2\right] \end{cases}$$

which is readily solved by substituting the first equation into the second. However, attempting to find a simplifying distribution that matches even the first three moments of the true distribution is not straightforward, as it requires the solution of a system of three nonlinear equations, including a cubic equation:

$$\begin{cases} \frac{1}{3}\left(x_1 + x_2 + x_3\right) = E\left[X\right] \\ \frac{1}{3}\left(x_1^{\,2} + x_2^{\,2} + x_3^{\,2}\right) = E\left[X^2\right] \\ \frac{1}{3}\left(x_1^{\,3} + x_2^{\,3} + x_3^{\,3}\right) = E\left[X^3\right] \end{cases}$$

Clearly the problem becomes even more difficult for polynomials of higher degree. Thus this approach, which seeks *n* support points of equal probability to match the first *n* moments, does not appear promising.

An alternate method is to fix the support points but vary the probability assigned to each; for instance, instead of solving the system of three nonlinear equations above, we would solve

$$\begin{cases} p_1 x_1 + p_2 x_2 + p_3 x_3 = E\left[X\right] \\ p_1 x_1^{\,2} + p_2 x_2^{\,2} + p_3 x_3^{\,2} = E\left[X^2\right] \\ p_1 x_1^{\,3} + p_2 x_2^{\,3} + p_3 x_3^{\,3} = E\left[X^3\right] \end{cases}$$

where $x_1$, $x_2$, and $x_3$ are constant and determined beforehand. Thus, no matter the degree of *n*, the system remains linear and allows a ready solution from linear algebra:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \\ x_1^3 & x_2^3 & x_3^3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} E\left[X\right] \\ E\left[X^2\right] \\ E\left[X^3\right] \end{bmatrix}$$

As a geometric progression is exhibited in the columns of the matrix (i.e., it is a Vandermonde matrix) the matrix is nonsingular (assuming the $x_i$ are distinct and nonzero) and a solution is readily attained. However, it is not at all guaranteed that the solution yields valid probabilities that are nonnegative and sum to one, nor is it clear how to determine the support points $x_i$ in the first place. Presumably allowing more than three support points increases the likelihood of finding a feasible solution, although it is difficult to make any general statements.

In the literature, this is known as the discrete moment problem. Prékopa (1990) proposes a linear programming formulation which adds constraints that the probabilities be nonnegative and sum to one, and produces results that make use of the Vandermonde coefficient matrix. However, as our algorithm has to solve this problem numerous times, a linear programming approach is undesirable from a computational standpoint.

Thus we adopt a different approach. Instead of deriving analytical expressions or numerical techniques for creating a simplifying distribution, it is possible to calculate the moments of a path if we know the distribution of the costs on its first arc, and the moments of the subpath from the second arc on to the destination. Let random variables $X$ and $C$ respectively denote the cost of

this subpath, and the cost of the arc we are adding to the subpath. Let $C_s$ and $p_s$, $s \in \{1, \ldots, S\}$, index the cost and probability for each possible state of this arc, and assume we know the moments $E[X^k]$ of the subpath for $k \in \{1, \ldots, n\}$. We can directly calculate the $n$-th moment of the random variable $X + C$ using the binomial theorem:

$$\begin{aligned} E_{X,C}\left[(X+C)^n\right] &= \sum_{s=1}^{S} p_s E_X\left[(X+C_s)^n \middle| C_s\right] \\ &= \sum_{s=1}^{S} p_s E_X\left[\sum_{k=0}^{n}\binom{n}{k} X^k (C_s)^{n-k} \middle| C_s\right] \\ &= \sum_{s=1}^{S}\sum_{k=0}^{n}\binom{n}{k} p_s (C_s)^{n-k} E\left[X^k \middle| C_s\right] \end{aligned} \qquad (4.1)$$

all of which we know by assumption. Since we can calculate the moments of $X + C$, we can calculate the expected utility from following this extended path:

**Theorem 4.1**. If $C$ and $X$ are random variables denoting the cost of an arc, and of a path adjacent to that arc, and if the utility function $P$ is a polynomial of degree $d$, then the expected utility of the path $C \cup X$ when departing at time zero is

$$E\left[P(X+C)\right] = \sum_{i=0}^{d} a_i E\left[(X+C)^i\right] = \sum_{i=0}^{d}\sum_{s=1}^{S}\sum_{k=0}^{i}\binom{i}{k} a_i p_s (C_s)^{i-k} E\left[X^k \middle| C_s\right]$$

**Proof**. Follows immediately from the expressions for the moments of $X + C$ and for calculating the expected utility from the moments of the cost distribution, both discussed above. QED

**Corollary 4.1**. If the conditions of Theorem 4.1 hold, except we are departing at any time $t$ (not necessarily zero), the expected utility of $C \cup X$ is:

$$E\left[P\left(t+X+C\right)\right]=\sum_{i=0}^{d}a_i E\left[\left(t+X+C\right)^i\right]=\sum_{i=0}^{d}\sum_{s=1}^{S}\sum_{k=0}^{i}\binom{i}{k}a_i p_s\left(C_s\right)^{i-k}E\left[\left(t+X\right)^k\middle|C_s\right]$$

**Proof**. Replace $X$ in Theorem 4.1 with a new random variable $X' = t + X$.

In Corollary 4.1, the distribution of $X'$ is that of arrival times at the destination; the distribution of $X$ is that of travel times from the current node to the destination. Naturally, they differ by a constant $t$ representing the current time.

It is possible to verify that the expressions in Corollary 3.2 are consistent with this theorem. For the case of deviance, we have $d = 2$, $\mathbf{a} = [(X^*)^2 \quad -2X^* \quad 1]$, so

$$\begin{aligned}
E\left[D\left(t+X+C\right)\right]&=\sum_{i=0}^{d}\sum_{s=1}^{S}\sum_{k=0}^{i}\binom{i}{k}a_i p_s\left(C_s\right)^{i-k}E\left[\left(t+X\right)^k\middle|C_s\right]\\
&=\sum_{s=1}^{S}p_s\left\{\begin{array}{l}\left(X^*\right)^2-2X^*\left[E\left[t+X\middle|C_s\right]+C_s\right]+\\ \left[E\left[\left(t+X\right)^2\middle|C_s\right]+2C_sE\left[t+X\middle|C_s\right]+C_s^{\,2}\right]\end{array}\right\}\\
&=\sum_{s=1}^{S}p_s\left\{\begin{array}{l}\left(X^*\right)^2-2X^*\left[E\left[X\middle|C_s\right]+t+C_s\right]+\\ \left[E\left[X^2+2Xt+t^2\middle|C_s\right]+2C_st+2C_sE\left[X\middle|C_s\right]+C_s^{\,2}\right]\end{array}\right\}\\
&=\sum_{s=1}^{S}p_s\left\{\begin{array}{l}-2X^*\left[E\left[X\middle|C_s\right]+t+C_s\right]+E\left[X^2+2C_sX+C_s^{\,2}\middle|C_s\right]\\ +2tE\left[X\middle|C_s\right]+2C_st+t^2+\left(X^*\right)^2\end{array}\right\}
\end{aligned}$$

$$
\begin{aligned}
E\left[D\left(t+X+C\right)\right] &= \sum_{s=1}^{S} p_s \left\{ \begin{array}{l} -2X^*t - 2X^*\left[E\left[X+C_s\middle|C_s\right]\right] + E\left[\left(X+C_s\right)^2\middle|C_s\right] \\ +2tE\left[X+C_s\middle|C_s\right] + 2C_sE\left[X\middle|C_s\right] + t^2 + \left(X^*\right)^2 \end{array} \right\} \\
&= \sum_{s=1}^{S} p_s \left\{ \begin{array}{l} E\left[\left(X+C_s\right)^2\middle|C_s\right] + t^2 + \left(X^*\right)^2 - 2X^*t \\ -2X^*\left[E\left[X+C_s\middle|C_s\right]\right] + 2tE\left[X+C_s\middle|C_s\right] \end{array} \right\} \\
&= \sum_{s=1}^{S} p_s \left\{ \begin{array}{l} V\left[X+C_s\middle|C_s\right] + \left(E\left[\left(X+C_s\right)\middle|C_s\right]\right)^2 + t^2 + \left(X^*\right)^2 \\ -2X^*t - 2X^*\left[E\left[X+C_s\middle|C_s\right]\right] + 2tE\left[X+C_s\middle|C_s\right] \end{array} \right\} \\
&= \sum_{s=1}^{S} p_s \left\{ V\left[X+C_s\middle|C_s\right] + \left(E\left[\left(X+C_s\right)\middle|C_s\right] - X^* + t\right)^2 \right\} \\
&= V\left[X+C\right] + \left(E\left[X+C\right] - \left(X^* - t\right)\right)^2
\end{aligned}
$$

Thus if we can show that $V[X + C]$ and $E[X + C]$ are equal to the values given in Corollary 3.2, we are done. $E[X + C]$ is easiest:

$$
E\left[X+C\right] = \sum_{s=1}^{S} p_s E\left[X+C_s\middle|C_s\right] = \sum_{s=1}^{S} p_s \left(C_s + E\left[X\middle|C_s\right]\right)
$$

Likewise, for variance, we have

$$
\begin{aligned}
V\left[X+C\right] &= E\left[\left(X+C\right)^2\right] - \left(E\left[X+C\right]\right)^2 \\
&= \left(\sum_{s=1}^{S} p_s E\left[\left(X+C_s\right)^2\middle|C_s\right]\right) - \left(\sum_{s=1}^{S} p_s\right)\left(E\left[X+C\right]\right)^2 \\
&= \sum_{s=1}^{S} p_s \left\{ E\left[X^2 + 2XC_s + C_s^2\middle|C_s\right] - \left(E\left[X+C\right]\right)^2 \right\} \\
&= \sum_{s=1}^{S} p_s \left\{ E\left[X^2\middle|C_s\right] + 2C_sE\left[X\middle|C_s\right] + C_s^2 - \left(E\left[X+C\right]\right)^2 \right\}
\end{aligned}
$$

$$
\begin{aligned}
V[X+C] &= \sum_{s=1}^{S} p_s \left\{ V[X|C_s] + \left(E[X|C_s]\right)^2 + 2C_s E[X|C_s] + C_s^2 - \left(E[X+C]\right)^2 \right\} \\
&= \sum_{s=1}^{S} p_s \begin{Bmatrix} V[X|C_s] + \left(E[X|C_s] + C_s - E[X+C]\right)^2 \\ -2\left(E[X+C]\right)^2 + 2E[X|C_s]E[X+C] + 2C_s E[X+C] \end{Bmatrix} \\
&= \sum_{s=1}^{S} p_s \begin{Bmatrix} V[X|C_s] + \left(E[X|C_s] + C_s - E[X+C]\right)^2 \\ -2E[X+C]\left(E[X+C] - E[X|C_s] - C_s\right) \end{Bmatrix} \\
&= \sum_{s=1}^{S} p_s \left\{ V[X|C_s] + \left(E[X|C_s] + C_s - E[X+C]\right)^2 \right\} - \\
&\quad 2E[X+C]\left( E[X+C] - \sum_{s=1}^{S} p_s \left(E[X|C_s] + C_s\right) \right) \\
&= \sum_{s=1}^{S} p_s \left\{ V[X|C_s] + \left(E[X|C_s] + C_s - E[X+C]\right)^2 \right\} - \\
&\quad 2E[X+C]\left(E[X+C] - E[X+C]\right) \\
&= \sum_{s=1}^{S} p_s \left\{ V[X|C_s] + \left(E[X|C_s] + C_s - E[X+C]\right)^2 \right\}
\end{aligned}
$$

which is consistent with Corollary 3.2, showing that, although the algebra is cumbersome, the equations developed for MDR-MC are a special case of Corollary 4.1.

## 4.4 Piecewise Polynomial Functions

Corollary 4.1 gives a formula for evaluating the utility of extending a path by one arc given the moments of the path cost distribution, which can in turn be calculated using (4.1). While this derivation was only accomplished for functions of one piece, it is straightforward to extend this to allow for functions of multiple pieces.

If $P(x) = \sum_{j=1}^{N} \sum_{i=0}^{d_j} a_{ji} x^i I_{x \in A_j}(x)$, so there are $N$ pieces (indexed by $j$), each of which has degree $d_j$, then it is enough to store the $D \equiv \sum_{j=1}^{N} d_j$ values $E\left[ x^i I_{x \in A_j}(x) \right]$ at each NTPC for $\{(i, j): j \in \{1, \ldots, N\}, i \in \{1, \ldots, d_j\}\}$. Essentially, we treat each piece separately, and determine its moments using an equation similar to (4.1). To avoid explicit calculation of the expected value of a product involving an indicator function, we introduce variables $\rho_j$ which represent the probability that the arrival time at the destination is contained in interval $A_j$, and calculate moments conditional on arriving in that interval:

$$\begin{aligned} E_{X,C}\left[ (X+C)^n I_{x \in A_j}(X+C+t) \right] &= \sum_{s=1}^{S} p_s E_X\left[ (X+C_s)^n I_{x \in A_j}(X+C_s+t) \middle| C_s \right] \\ &\sum_{s=1}^{S} p_s \rho_{j|C_s} E_X\left[ (X+C_s)^n \middle| C_s, A_j \right] \\ &= \sum_{s=1}^{S} p_s \rho_{j|C_s} E_X\left[ \sum_{k=0}^{n} \binom{n}{k} X^k (C_s)^{n-k} \middle| C_s, A_j \right] \\ &= \sum_{s=1}^{S} \sum_{k=0}^{n} \binom{n}{k} p_s \rho_{j|C_s} (C_s)^{n-k} E\left[ X^k \middle| C_s, A_j \right] \end{aligned} \tag{4.2}$$

where $\rho_{j|C_s}$ is the probability of arriving in interval $A_j$ given that the current arc is in state $s$. We calculate $\rho_j$ using the equation $\rho_j = \sum_{s=1}^{S} p_s \rho_{j|C_s}$. Likewise we have

$$E\left[ (X+C)^k \middle| A_j \right] = \frac{E\left[ (X+C)^n I_{x \in A_j}(X+c+t) \right]}{\rho_j} \tag{4.3}$$

defining the quotient to be zero if $\rho_j = 0$.

Thus, at each NTPC, for each piece $j$, we store a variable $\rho_j$ denoting the probability of arriving at the destination in interval $A_j$, as well as the $d_j$ moments of the remaining travel cost conditional on arriving in that interval.

We now state Corollary 4.2 without proof, as the proof is identical to Corollary 4.1, although the notation is more cumbersome.

**Corollary 4.2**. If we are departing at any time $t$, the expected utility of $C \cup X$ is:

$$E\left[P\left(t+X+C\right)\right] = \sum_{j=1}^{N}\sum_{i=0}^{d}\sum_{s=1}^{S}\sum_{k=0}^{i}\binom{i}{k} a_{ji}\, p_s \rho_j \left(C_s\right)^{i-k} E\left[\left(t+X\right)^k \middle| C_s, A_j\right]$$

## 4.5 Algorithm MPPR-MC

With the model and utility function determined, and with formulas to enable a solution to the optimal policy problem via backward induction, we now describe an algorithm to do just that. Pseudocode is presented in Section 4.5.1; proofs of correctness in acyclic networks are given in Section 4.5.2; bounds for cyclic networks are discussed in Section 4.5.3; and complexity results are found in Section 4.5.4. A fully worked-out example of this algorithm applied to a small network is found in Appendix B.

Initialization.

$\forall$ $a,s,t,j,n$ with $a \in RS(D)$, $s \in S(a)$, $t \in T(D)$, $j \in \{1, \ldots, N\}$, $n \in \{1, \ldots, d_j\}$

$L(D,t,a,s,j,n) := 0$

$L(D,t,a,s,j,0) := 1$

$\rho(i,t,a,s,j) := 1$ if $t \in A_j$, 0 otherwise.

$\forall$ $i,t,a,s,j,n$ with $i \in (N \setminus D)$, $t \in T(j)$, $a \in RS(i)$, $s \in S(a)$, $j \in \{1, \ldots, N\}$, $n \in \{1, \ldots, d_j\}$

$L(i,t,a,s,j,n) := \infty$,

$\rho(i,t,a,s,j) := \infty$.

$SEL := \{(i,t): i \in \Gamma^{-1}(D), t \in T(i)\}$

Iteration:

while $SEL \neq \emptyset$

Select $(i,t) \in SEL$ and set $SEL := SEL \setminus (i,t)$

For $\forall$ $a,b,s$, with $a \in RS(i)$, $b = (i,q) \in FS(i)$, $s \in S(a,t)$

For $j \in \{1, \ldots, N\}$, $n \in \{1, \ldots, d_j\}$

$$\rho_b(i,t,a,s,j) := \sum_{s \in S(b)} p(b,k,t,a,s)\rho(q,t+c_s,b,s,j)$$

$$L_b(i,t,a,s,j,n) := \sum_{s \in S(b)} \sum_{k=0}^{n} \binom{n}{k} \frac{p_s}{\rho_b(i,t,a,s,j)} \rho(q,t+c_s,b,s,j)(c_s)^{n-k} L(q,t+c_s,b,s,j,k)$$

$$P_b(i,t,a,s) := \sum_{j=1}^{N} \sum_{n=0}^{d} \sum_{k=0}^{n} \binom{n}{k} a_{ji} \rho_b(i,t,a,s,j) t^{n-k} L_b(i,t,a,s,j,k)$$

if $P_b(i,t,a,s) < P(i,t,a,s)$

For $j \in \{1, \ldots, N\}$, $n \in \{1, \ldots, d_j\}$

$L(i,t,a,s,j,n) := L_b(i,t,a,s,j,n)$

$\rho(i,t,a,s,j) := \rho_b(i,t,a,s,j)$

$P(i,t,a,s) := P_b(i,t,a,s)$

$\pi(i,t,a,s) := b$

$SEL := SEL \cup \{(h,u): h \in \Gamma^{-1}(i), u = t - c^{r}_{(h,i)}$ for some $r \in \{1, 2, \ldots, S((h,i), t - c^{r}_{(h,i)})\}$

**Figure 4.1**. Pseudocode for Algorithm MPPR-MC

### *4.5.1 Pseudocode for Algorithm MPPR-MC*

The operation of algorithm MPPR-MC is very similar to that of MDR-MC, although more labels need to be computed at each NTPC. Let $L(i, t, a, s, j, n)$

denote the label stored at node *i*, at time *t*, when arriving via arc *a* in state *s*, that represents the *n*-th moment of remaining travel time conditional on arrival during $A_j$, $E\left[X^n \middle| C_s, A_j\right]$, and let $\rho(i,t,a,s,j)$ denote the probability of arrival during $A_j$ for the same NTPC. Let $P(i, t, a, s)$ denote the expected disutility of travel when departing that NTPC. As before, $\pi$ denotes the path pointer. To calculate expected disutility using Corollary 4.2, we need to calculate $E\left[(X+t)^k \middle| C_s, A_j\right]$ from the labels $E\left[X^k \middle| C_s, A_j\right]$. This is accomplished with a straightforward application of the binomial theorem:

$$E\left[(X+t)^n \middle| C_s, A_j\right] = E\left[\sum_{k=0}^{n}\binom{n}{k}X^k t^{n-k} \middle| C_s, A_j\right] = \sum_{k=0}^{n}\binom{n}{k}t^{n-k}E\left[X^k \middle| C_s, A_j\right]$$

A fully worked-out example of this algorithm applied to a small network is found in Appendix B.

*4.5.2 Proofs of Correctness*

These proofs, given for acyclic networks, are nearly identical to those given in Section 3.4.3. Complications arising in cyclic networks are discussed in Section 4.5.3. The notation used here is the same as in Section 3.4.3.

---

**Theorem 4.2**. In acyclic networks, algorithm MPPR-MC terminates after finitely many iterations.

**Proof**. By contradiction, assume that the algorithm never terminates. This implies that *SEL* is never empty. Since an element is removed from *SEL* at each iteration, if it is never empty there must be infinitely many additions to *SEL*. Since the number of node-time pairs in such networks is finite, some node-time pair must be added to *SEL* infinitely many times; however, this contradicts the assumption that the network is acyclic, since the number of times a node-time pair is scanned is bounded by the maximum number of arcs that can be traveled before the destination is reached, which is finite. QED

---

**Lemma 4.1**. Upon termination of the algorithm, we have a policy $\pi$ that minimizes $P$ at each $\phi \in \Phi$.

**Proof**. By contradiction, assume that the algorithm has terminated, but there exists $\phi \in \Phi$ such that the resulting policy does *not* minimize $P$ there; that is, there exist arcs $b$, $c$ in $FS(\phi_n)$ such that $\pi(\phi) = b$ and $P_b^\pi(\phi) > P_b^\pi(\phi)$. Let $k$ be the node reached by following arc $c$. Since a path pointer exists for $(\phi_n, \phi_t)$ it must have been in *SEL* at least once; consider the last time that this pair was chosen from *SEL*. Since successor node $b$ was instead of $c$, we know $P_b^\pi(\phi) \leq P_b^\pi(\phi)$ when $i$

was last scanned. However, at termination, by assumption we have $P_b^{\pi}(\phi) > P_b^{\pi}(\phi)$, indicating that $P_b^{\pi}(\phi)$ must have changed again before termination, which implies that some $P$ label at a node-time pair $(k, \phi_t + c_{c,\phi_t}^{r})$ has changed for some $r \in \{1, 2, \ldots, S(c, \phi_t)\}$. But if this happened, $(\phi_n, \phi_t)$ would reenter SEL, contradicting the assumption that this was the last time $(\phi_n, \phi_t)$ was chosen from *SEL*. Thus, when the algorithm terminates, the resulting policy minimizes $P$ for each $\phi$ in $\Phi$.

---

**Lemma 4.2**. For any $\phi_0 \in \Phi$ and arc $b \in FS(\phi_0)_n$, and any two policies $A$ and $B$ for which $P^A(\phi) = P^B(\phi)$ for all $\phi \in \Phi^{+1}{}_b(\phi_0)$, we have $P_b^A(\phi_0) = P_b^B(\phi_0)$.

**Proof**. The disutility label at any NTPC $\psi$ under a policy $\pi$ can be expressed as $\sum_{\omega \in \Omega} p_{\omega,\pi|\psi} P(X_\omega)$. Therefore, we have

$$P_b^A(\phi_0) = \sum_{\omega} p_{\omega,A|\phi_0} P(X_\omega) = \sum_{\omega} \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} p_{\omega,A|\phi} P(X_\omega) =$$
$$= \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} \sum_{\omega} p_{\omega,A|\phi} P(X_\omega) = \sum_{\phi \in \Phi_b^{+1}(\phi_0)} \rho_{\phi,b|\phi_0} \sum_{\omega} p_{\omega,B|\phi} P(X_\omega) = P_b^B(\phi_0)$$

since $\sum_{\omega} p_{\omega,A|\phi} P(X_\omega) = \sum_{\omega} p_{\omega,B|\phi} P(X_\omega)$ for all $\phi \in \Phi^{+1}{}_b(\phi_0)$ by assumption.

---

**Lemma 4.3**. All policies that minimize $P(\phi)$ at all $\phi \in \Phi$ have the same $P(\psi)$ for all $\psi \in \Phi$, and, in particular, the same $P(\mathbb{O})$.

**Proof**. Let $A$ and $B$ be two distinct policies that minimize $P$ at each NTPC. (The claim is trivial if the set of such policies is a singleton). Let $\Phi_K$ denote the set of NTPCs that are at most $K$ arcs away from the destination. Since the network is acyclic, each NTPC belongs to some $\Phi_K$. We now show that $P^A(\phi) = P^B(\phi)$ for all $\phi \in \Phi$ by strong induction. Consider any $\phi_1 \in \Phi_1$. Since $\Phi^{+1}(\phi_1) \subseteq \underline{\mathbb{D}}$ and clearly $P^A(\phi) = P^B(\phi)$ for all $\phi \in \underline{\mathbb{D}}$, $P^A(\phi) = P^B(\phi)$) by Lemma 3.3. Now assume that $P^A(\phi) = P^B(\phi)$ for all $\phi \in \bigcup_{k=1}^{I} \Phi_k$. Consider any $\phi_{I+1} \in \Phi_{I+1}$. By the induction hypothesis $P^A(\phi) = P^B(\phi)$ for $\phi \in \Phi^{+1}(\phi_{I+1}) \subseteq \bigcup_{k=1}^{I} \Phi_k$, so by Lemma 3.3 we also have $P^A(\phi) = P^B(\phi)$. Thus, for all $K$, $P^A(\phi) = P^B(\phi)$ for $\phi \in \Phi_K$. Since each $\phi \in \Phi$ belongs to some $\Phi_K$, this proves the claim.

---

**Lemma 4.4**. Any policy that does not minimize $P(\phi)$ for all $\phi \in \Phi$ reached with positive probability is suboptimal.

**Proof**. Let $A$ be a policy as described above. Then there exist some $\phi^* = (i, t, a, s)$ and arcs $b$, $c$ such that $A(\phi) = b$ but $P_b^A(\phi) > P_c^A(\phi)$. Define a new policy $B$ that is identical to $A$ except $B(\phi) = c$. We now show that $P^B(\mathbb{O}) < P^A(\mathbb{O})$, thereby proving the lemma.

Recall that $\Omega$ is the set of all realizations of the network. Partition $\Omega$ into sets $\Pi_1$ and $\Pi_2$ where $\Pi_1$ contains all realizations containing $\phi^*$, and $\Pi_2 = \Omega \setminus \Pi_1$. Thus, we have

$$P^A(\mathbb{O}) = \sum_{\omega \in \Pi_1} p_{\omega,A} P(X_\omega) + \sum_{\omega \in \Pi_2} p_{\omega,A} P(X_\omega)$$

$$P^B(\mathbb{O}) = \sum_{\omega \in \Pi_1} p_{\omega,B} P(X_\omega) + \sum_{\omega \in \Pi_2} p_{\omega,B} P(X_\omega)$$

Since $A$ and $B$ are identical except at $\phi^*$, $p_{\omega,A} = p_{\omega,B}$ for all $\omega \in \Pi_2$, so

$$P^A(\mathbb{O}) - P^B(\mathbb{O}) = \sum_{\omega \in \Pi_1} p_{\omega,A} P(X_\omega) - \sum_{\omega \in \Pi_1} p_{\omega,B} P(X_\omega)$$

Since $A$ and $B$ differ only at $\phi^*$, we have $\sum_{\omega \in \Pi_1} p_{\omega,A} = \sum_{\omega \in \Pi_1} p_{\omega,B}$, and by assumption both sums are strictly positive. Thus we divide the first term by $\sum_{\omega \in \Pi_1} p_{\omega,A}$ and the second by $\sum_{\omega \in \Pi_1} p_{\omega,B}$, which yields

$$\frac{\sum_{\omega \in \Pi_1} p_{\omega,A} P(X_\omega)}{\sum_{\omega \in \Pi_1} p_{\omega,A}} - \frac{\sum_{\omega \in \Pi_1} p_{\omega,B} P(X_\omega)}{\sum_{\omega \in \Pi_1} p_{\omega,B}} =$$

$$= \sum_{\omega \in \Pi_1} p_{\omega,A|\phi^*} P(X_\omega) - \sum_{\omega \in \Pi_1} p_{\omega,B|\phi^*} P(X_\omega) =$$

$$= P_b^A(\phi^*) - P_c^A(\phi^*)$$

since $A$ and $A'$ differ only at *NTPC*. But $P_b^A(\phi^*) > P_c^A(\phi^*)$ so $P^A(\mathbb{O}) - P^B(\mathbb{O}) > 0$. Thus $P^A(\mathbb{O}) > P^B(\mathbb{O})$, completing the proof that $A$ is suboptimal.

---

**Theorem 4.3**. Any policy produced by algorithm MPPR-MC is optimal.

**Proof**. Let the set of all possible policies be partitioned into sets $M$ and $N$, where $M$ consists of all policies that minimize $P$ at all NTPCs, and $N$ consists of all other policies. Since there are only finitely many policies, there must be some policy whose deviance is no greater than that of any other policy. By Lemma 4.4, no policy in set $N$ is optimal; thus, some policy in $M$ is optimal. By Lemma 4.3, all policies in $M$ have the same deviance; hence all policies in $M$ are optimal. By Lemma 4.1, the algorithm will produce a policy in $M$; thus it will return an optimal policy. QED

---

**Corollary 4.3 (Bellman's Principle)**. If $A$ is an optimal policy for reaching the destination from origin $\mathbb{O}_1 = \phi_1 \in \Phi$, then for any $\phi_2 \in \Phi^+(\phi_1, A)$, the policy $B$ with $B(\phi) = A(\phi)$ for all $\phi \in \Phi^+(\phi_2, A)$ is optimal for reaching the destination from origin $\mathbb{O}_2 = \phi_2$.

**Proof**. Assume not. Then there exists an optimal policy $C$ such that $P^C(\phi_2) < P^B(\phi_2)$. From Lemma 4.4, for all $\phi \in \Phi^+(\phi_2,A)$, $C(\phi) \in \arg\min_{i \in FS(\phi)} P_i(\phi)$. Since $B$ is suboptimal, there exists some $\phi^* \in \Phi^+(\phi_2,A)$ for which $B(\phi) \neq C(\phi)$, and furthermore, by Lemma 4.3, for which $B(\phi^*) \notin \arg\min_{i \in FS(\phi)} P_i(\phi^*)$. However since $A(\phi) = B(\phi)$ for all $\phi \in \Phi^+(\phi_2, A)$ this implies $A(\phi^*) \notin \arg\min_{i \in FS(\phi)} P_i(\phi^*)$, so, by Lemma 4.4, $A$ is suboptimal, contradicting the original assumption. Thus no such policy $C$ exists, so $B$ is itself optimal.

---

*4.5.3 Complications in Cyclic Networks*

As discussed in Section 3.4.4, the presence of cycles in a network can be problematic, because an optimal policy may contain infinitely many cycles (albeit with vanishing probability). Even if the optimal policy contains no cycles, the time-expanded graph becomes infinite and algorithm MPPR-MC will not terminate.

As with MDR-MC, the key to avoiding these difficulties is to constrain the policy to reach the destination by some time $T$, and forming a bound the suboptimality induced by this constraint, since the probability of reaching the destination beyond time $T$ in the optimal policy is small, when $T$ is large.

However, unlike MDR-MC, with a piecewise polynomial utility function $P(x)$, there is no guarantee that $P$ is increasing for sufficiently large $x$. Indeed, if the coefficient of the term of largest degree in the last piece $P$ is negative, the optimal strategy will be to cycle indefinitely, and the problem is unbounded and has no solution. Lemma 4.5 ensures that we only need to consider one piece of the function when arrival times are sufficiently large:

---

**Lemma 4.5**. If a cycle exists in $G$, there exist $j \in \{1, \ldots, N\}$ and $x_0 \in \mathbb{R}$ such that $x \in A_j$ whenever $x > x_0$.

**Proof**. Since a cycle exists, the set of arrival times at the destination is infinite. By assumption each possible arrival time is contained in some $A_j$. Define $M_j = \sup\{x: x \in A_j\}$ (possibly $\infty$). Since the $A_j$ are intervals, they are connected; since they are also disjoint, the $M_j$ can be ordered; since the $A_j$ have positive measure, this ordering is strict. Without loss of generality let $M_1 < M_2 < \ldots < M_N$. Now define $x_0 = M_{N-1}$ and consider any $x > x_0$. Since $x > M_j$ for all $j$ except $N$, $x$ must belong to $A_N$. Thus all $x > x_0$ belong to the same piece of $P$.

---

Thus we need only consider one piece of the disutility function when concerned with large arrival times; without loss of generality let this be piece $N$,

denoted by the function $P_N$. We state an assumption which eliminates the possibility of an unbounded solution.

**Assumption 4.1**. If a cycle exists in $G$, the coefficient $a_{N,d(N)} > 0$.

Under Assumption 4.1, $P$ is increasing for sufficiently large $x$. Define $\Xi = \max\left\{0, \inf\left\{x : x \in A_N\right\}, \sup\left\{x : \frac{dP}{dx} < 0\right\}\right\}$, so $P$ is increasing for $x > \Xi$. As with MDR-MC, we start by bounding the probability that the optimal policy completes a trip after time $\tau$, for $\tau > \Xi + Mn$:

**Lemma 4.6**. For $\tau > \Xi + Mn$, the probability that the traveler will complete a trip after time $\tau$ when following an optimal policy $\pi^*$ is no greater than $\frac{P_N(\Xi + Mn)}{P_N(\tau)}$.

**Proof.** Consider the following $\tau$-feasible policy: travel deterministically from the origin to any node that is part of a cycle, then traverse this cycle repeatedly until the time is greater than $\Xi$, at which point any acyclic path is followed to the destination. Since the first part of this trip has cost no greater than $\Xi + M$, and the second part has cost no greater than $M(n-1)$, this policy will result in arrival at the destination between times $\Xi$ and $\Xi + Mn$ and, thus, will have disutility no

greater than $P_N(\Xi + Mn)$. Therefore, the optimal policy can certainly have disutility no greater than $P_N(\Xi + Mn)$. Therefore we have

$$\sum_{\omega} p_{\omega,\pi^*} P\left(X^{\omega}\right) = \sum_{\omega: X^{\omega} \le \tau} p_{\omega,\pi^*} P\left(X^{\omega}\right) + \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*} P_N\left(X^{\omega}\right) \le P_N(\Xi + Mn)$$

$$\Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*} P_N\left(X^{\omega}\right) \le P_N(\Xi + Mn) \Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*} P_N(\tau) \le P_N(\Xi + Mn) \Rightarrow$$

$$\Rightarrow \sum_{\omega: X^{\omega} > \tau} p_{\omega,\pi^*} \le \frac{P_N(\Xi + Mn)}{P_N(\tau)}$$

proving the lemma.

---

Next, for a given $T > \Xi + Mn$, we bound the increase in disutility created by forcing a trip to end before $T$; as before, this is accomplished by constructing a (possibly suboptimal) $T$-feasible strategy $\pi^T$, where $\pi^T(\phi) = \pi^*(\phi)$ for $\phi$ such that $(\phi)_t \le T - Mn$; when $(\phi)_t > T - Mn$ we have the traveler deterministically follow any acyclic path to the destination. That is, the optimal unconstrained policy is used until time $T - Mn$, after which a deterministic path is followed. Such a policy is $T$-feasible since one can always reach the destination from any node incurring cost less than $Mn$.

---

**Theorem 4.4**. The difference between the disutility of an optimal policy and an optimal $T$-feasible policy is no greater than $P_N(\Xi + Mn)\left[\frac{P_N(T)}{P_N(T - Mn)} - 1\right]$

**Proof.** Partition Ω into two sets $\Omega_1$ and $\Omega_2$ with $\Omega_1 = \{\omega: X^\omega \geq T - Mn\}$ and $\Omega_2 = \{\omega: X^\omega < T - Mn\}$. Thus

$$P\left(\pi^*\right) = \sum_{\omega \in \Omega_1} p_{\omega,\pi^*} P\left(X^\omega\right) + \sum_{\omega \in \Omega_2} p_{\omega,\pi^*} P\left(X^\omega\right)$$

and

$$P\left(\pi^T\right) \leq P(T) \sum_{\omega \in \Omega_1} p_{\omega,\pi^*} + \sum_{\omega \in \Omega_2} p_{\omega,\pi^*} P\left(X^\omega\right)$$

since $p_{\omega,\pi^*} = p_{\omega,\pi^T}$ for $\omega \in \Omega_2$. Therefore

$$P(\pi^{T*}) - P(\pi^*) \leq P(\pi^T) - P(\pi^*) \leq \sum_{\omega \in \Omega_1} p_{\omega,\pi^*} \left[ P_N(T) - P_N\left(X^\omega\right) \right] \leq$$

$$\leq \sum_{\omega: X^\omega \in [T-Mn,T]} p_{\omega,\pi^*} \left[ P_N(T) - P_N\left(X^\omega\right) \right]$$

the last inequality because $P_N(T) - P_N\left(X^\omega\right)$ is negative when $X^\omega > T$. Therefore we have

$$P(\pi^{T*}) - P(\pi^*) \leq \sum_{\omega: X^\omega \in [T-Mn,T]} p_{\omega,\pi^*} \left[ P_N(T) - P_N\left(X^\omega\right) \right] \leq$$

$$\sum_{\omega: X^\omega \in [T-Mn,T]} p_{\omega,\pi^*} \left[ P_N(T) - P_N\left(T - M_N\right)^2 \right] \leq \frac{P_N(\Xi + Mn)}{P_N(T - M_N)} \left[ P_N(T) - P_N\left(T - M_N\right)^2 \right]$$

using Lemma 3.6, or $P(\pi^{T*}) - P(\pi^*) \leq P_N(\Xi + Mn) \left[ \frac{P_N(T)}{P_N(T - Mn)} - 1 \right]$ QED

Although this bound depends on the function $P_N$, one can use l'Hospital's Rule to show that this bound becomes arbitrarily small as $T \to \infty$. Derivation of an exact minimum value of $T$ given some desired tolerance $\varepsilon$, analogous to Corollary 3.4, is difficult in general; in fact, if $P_N$ has degree five or greater, the Abel-Ruffini Theorem excludes the possibility of an analytical solution.

However, it is not difficult to evaluate the bound in Theorem 4.4 numerically, choosing various values of $T$ until the bound is sufficiently small. Numerical calculation of $\Xi$ is not difficult either, and can be done by finding the largest root of the first derivative of $P_N$.

Also, if $d_N \geq 2$, for sufficiently small $\varepsilon$ we may use the bound in Corollary 3.4 which was derived for MDR-MC, because we have this inequality for large $T$:

$$1 \leq \frac{P_N(T)}{P_N(T - Mn)} \leq \frac{D(T, X^*)}{D(T - Mn, X^*)}$$

where $D(\bullet, X^*)$ represents deviance calculated from a target of $X^*$. This inequality holds because $P_N$, having degree no less than $D$, approaches its limiting value faster than $D$, at least for large $T$. However, it is not at all clear how large $T$ must be for this inequality to hold.

For the case when $d_N = 1$ (that is, the last piece is linear), we can express $P_N$ as $ax + b$. Then, analytically, we can write $P_N(\Xi + Mn)\left[\frac{P_N(T)}{P_N(T - Mn)} - 1\right]$ as $\frac{a^2M^2n^2 + a^2MnL + aMnb}{aT - aMn + b}$ where $\Xi = L = \inf\{x: x \in A_N\}$, and we can solve $\varepsilon < \frac{a^2M^2n^2 + a^2MnL + aMnb}{aT - aMn + b}$ for $\varepsilon$:

$$T > aMn - b + \frac{a^2M^2n^2 + a^2MnL + aMnb}{\varepsilon}$$

although we also require $T > L + Mn$ for the above results to hold.

As with the bounds developed in Section 3.4.4, these are likely to be quite loose in practice; however, it is still encouraging that a bound exists for any network, and any piecewise polynomial utility function satisfying Assumption 4.1.

*4.5.4 Complexity*

Again, let $T = \max_{\phi \in \Phi}\{\phi_t\}$ and $T = \max_{a \in A, t \in T(a)}\{|S(a,t)|\}$ refer to the latest arrival time at any node and the maximum number of states an arc exists in, respectively. Let $D$ be the largest degree of any of the polynomial pieces. In cyclic graphs assume we are solving a $T$-constrained problem. Recall that $M$ is the largest possible cost of any arc. The following proposition demonstrates that

MPPR-MC has pseudopolynomial complexity in both acyclic and cyclic graphs when the tolerance $\varepsilon$ is fixed.

---

**Proposition 4.1**. Algorithm MPPR-MC has complexity $O(nmNTD^2S^2min\{n,T\})$ when a FIFO SEL is used.

**Proof**. This proof of complexity mirrors that of the label-correcting deterministic shortest path algorithm presented in Ahuja et al (1993). With a FIFO SEL, at most min{$n$, $T$} passes are required; at worst, a summation must be made for each arrival time at the head of each arc, for each predecessor of this arc, and for each state this predecessor, for each of $D$ moments to be calculated for $N$ polynomials. each requiring. With $m$ arcs, at most $T$ arrival times for each node, at most $n$ predecessor arcs, at most $S$ states for any arc, and at most $DN$ labels, each pass requires at most $nmTSDN$ summations, each of which involves at most $SD$ terms. Hence each pass requires at most $nmNTD^2S^2$ calculations, for a total complexity of $O(nmNTD^2S^2min\{n,T\})$.

---

**Proposition 4.2**. For acyclic graphs, algorithm MPPR-MC has complexity $O(NMn^3mD^2S^2)$.

**Proof**. In acyclic networks, $T \leq Mn$, leading to a complexity of $O(NMn^3mD^2S^2)$.

---

**Proposition 4.3**. In networks with cycles, for a tolerance $\varepsilon > 0$, algorithm MPPR-MC determines a policy within $\varepsilon$ of the optimal policy with complexity $O(NM^3n^5mD^2S^2/\varepsilon)$ as $\varepsilon \to 0$, if $d_N \geq 2$. If $d_N = 1$, the complexity is $O(NM^2n^4mD^2S^2/\varepsilon)$.

**Proof**. As discussed above, when $d_N \geq 2$, the bound in Corollary 3.4 is also an upper bound for the $T$ needed to reduce error below $\varepsilon$, when $\varepsilon$ is sufficiently small. Likewise, for $d_N = 1$, we can express the upper bound for $T$ analytically, as was done above. Substitution into Proposition 4.1 completes the proof.

## 4.6 Approximations to Nonpolynomial Functions

Although Algorithm MPPR-MC is general in that it finds an optimal policy for an arbitrary piecewise polynomial function, there may be cases where we wish to use a nonpolynomial disutility function, such as an exponential or logarithmic function.

One can approximate some functions with a Taylor polynomial and then apply Algorithm MPPR-MC. Recall Taylor's Theorem:

$$f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!}(x-a)^i + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}$$

for some $\xi \in [a, x]$ when $f$ is $n + 1$ times continuously differentiable on $[a, x]$; here the remainder term takes the Lagrange form.

Let us consider some nonpolynomial function $f$ which is analytic in an interval containing all possible arrival times at the destination. We can write

$$E\left[ f(x) \right] = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!} E\left[ (x-a)^i \right] + E\left[ \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-a)^{n+1} \right]$$

If we approximate $f$ with the first $n + 1$ terms of its Taylor series (including the constant term $f(a)$), we will have a polynomial of degree $n$. We let $a$ represent the earliest possible arrival time at the destination. If $T$ is the latest possible arrival time at the destination (if cycles exist, assume the graph is $T$-constrained), then the expected error in this approximation is the expected value of the remainder term:

$$E\left[ \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-a)^{n+1} \right] \leq \frac{f^{(n+1)}(\eta)}{(n+1)!} (T-a)^{n+1}$$

where $\eta = \max\{ f^{(n+1)}(\xi) : \xi \in [a, T] \}$.

Since $f$ is analytic, we are ensured that the remainder term is arbitrarily small for sufficiently large $n$; the necessary size of $n$ can be determined by the bound given above.

For example, say $f(x) = \exp(kx)$. Then the error in approximating $f$ with a polynomial of degree $n$ is at most

$$\frac{f^{(n+1)}(\eta)}{(n+1)!}(T-a)^{n+1} = \frac{k^{n+1}e^{T}}{(n+1)!}(T-a)^{n+1} = e^{T}\frac{\left(k(T-a)\right)^{n+1}}{(n+1)!}$$

If $k$ is much larger than $(T-a)^{-1}$, this error can be quite large, because the rapid growth of the exponential function results in a poor polynomial approximation at the high end, at least until $n$ is itself quite large. However, this rapid growth in the disutility function suggests that relatively few trips will terminate in the range where the approximation is poorest, so it is quite possible that this bound is very conservative in practice.

## 4.7 Conclusion

In this chapter we developed a general algorithm MPPR-MC which finds an optimal routing policy in stochastic, time-varying networks with Markovian arc costs when a traveler's disutility is defined by a piecewise polynomial function. Correctness and complexity results were proved, and complications in cyclic networks were addressed. Approximation of nonpolynomial functions using Taylor series was also discussed, and a bound on the error associated with this procedure was found.

This algorithm generalizes much previous work in routing with recourse. For instance, it was shown that algorithm MDR-MC, presented in Chapter 3, is a

special case of MPPR-MC. Finding an online, time-dependent shortest path can be accomplished with MPPR-MC by choosing the disutility function $P(x) = x$. Minimizing a linear combination of expected travel time, expected early arrival, and expected late arrival, as was done in Gao (2005), can also be accomplished using this algorithm.

The existence of a general algorithm for a broad class of utility functions can be very useful in tailoring routing algorithms to specific applications, such as delivery of perishable goods, where a single linear function cannot adequately capture the preferences of the traveler.

# Chapter 5: Numerical Analysis

## 5.1 Introduction

Although Chapters 3 and 4 prove correctness and worst-case complexity results for algorithms MDR-MC and MPPR-MC, the practical application of these algorithms is also of interest. In this chapter we examine how computation time varies with the problem size in randomly generated networks, to compare with the worst-case estimates obtained in the previous chapters.

Additionally, for MDR-MC, we investigate the quality of the solution in greater depth, examining the tradeoff between reliability and expected travel cost by comparing the minimum deviance policy to the minimum expected value policy. Comparison is also made between online and offline solution of the minimum deviance problem, to quantify the benefit of allowing travelers to update their decisions *en route*. No such in-depth analysis is performed for MPPR-MC, as the solution characteristics depend greatly on the choice of polynomial.

In Section 5.2, we discuss how random networks were generated. Section 5.3 describes the implementation of algorithms MDR-MC and MPPR-MC. Section 5.4 gives results on computation time requirements, while Section 5.5

compares the minimum deviance solution to an online shortest path solution. Section 5.6 investigates the benefit of finding a recourse solution, as opposed to an offline one, and Section 5.7 summarizes these findings.

## 5.2 Network Generation

To evaluate these algorithms, a large number of random networks of various sizes and structure were created. This section describes the procedure used to generate these networks.

For MDR-MC, the required parameters are the desired number of nodes, the average node connectivity (implicitly giving a desired number of arcs), the number of states each arc can exist in, the minimum and maximum arc costs, whether cycles are desired or not, and the type of correlation structure desired. Additionally, for MPPR-MC, we also require the number of polynomial pieces, and the degree of these pieces, to be specified.

First, the specified number of nodes $n$ is created and labeled 0 to $n$, and connectivity is ensured by creating a Hamiltonian path, which is formed by creating an arc connecting each successive pair of nodes $(i, i + 1)$. Subsequent arcs are generated by randomly generating the head and tail nodes. If an acyclic graph is specified, we require the head node to have a greater label than the tail

node; in addition to ensuring an acyclic graph, this allows the node labels to form a topological ordering. Arc costs are randomly generated using a discrete uniform distribution between the specified minimum and maximum arc costs, and ordered such that higher costs correspond to larger state indices. Although the algorithms developed can accommodate time-dependent arc costs, for these numerical investigations we assume that the distributions do not change with time. This procedure is repeated until the number of arcs created satisfies the input average node connectivity.

After generating all arcs, we specify the probabilities that each arc exists in a particular state. Since these depend on the state of the previous arc, we must wait until all arcs have been created before assigning probabilities. In this work, three methods are used to assign these probabilities, depending on the desired correlation structure: positive, independent, or negative.

Positive correlation implies that if one arc is has a high (or low) cost, so will its successor. To model this, we assume that the probability that an arc is in a state $s$ decreases with the difference between $s$ and the index of the previous arc state $r$. (This is valid since the states are ordered.) The particular equation used to find these probabilities is:

$$p(s|r) \propto S - |s-r| \Leftrightarrow p(s|r) = \frac{S - |s-r|}{\sum_{s=1}^{S} (S - |s-r|)}$$

For the independent case, we assign equal probability to all states regardless of the state of the previous arc:

$$p(s|r) = \frac{1}{S}$$

To model negative correlation, where a low cost in one arc suggests a higher cost in the next arc, and vice versa, we choose probabilities that *increase* in $|r-s|$:

$$p(s|r) = \frac{1 + |s-r|}{\sum_{s=1}^{S} (1 + |s-r|)}$$

Although we do not typically expect real transportation networks to exhibit negative correlation, we nevertheless develop this structure to compare the solution characteristics with the positive correlation and independent cases, which more closely resemble real networks.

To summarize, random networks are generated by following these steps:

1. Create the specified number of nodes.
2. Create a Hamiltonian path to ensure connectivity.
3. Generate additional arcs to satisfy desired average node connectivity.

4. Generate arc costs randomly.

5. Generate probability matrices using the desired correlation structure.

## 5.3 Implementation

The pseudocode given in Figures 3.4 and 4.1 does not specify how node-time pairs should be selected from the scan eligible list. Various implementations of this structure exist; for instance, Ziliaskopoulos and Mahmassani (1993) compare a FIFO queue to a deque implementation suggested by Pape (1974).

All of the networks tested in this chapter were acyclic, so an efficient implementation of the scan eligible list is to traverse the nodes in reverse topological order, examining all NTPCs corresponding to this node before scanning the previous node. In cyclic networks, an alternative method is to adapt algorithm DOT (Chabini, 1998) by starting at the latest arrival time $T$ and moving to progressively earlier times, scanning all NTPCs corresponding to each time interval; this procedure is also efficient because arc costs are assumed to be positive, so labels never need to be updated once set.

Algorithms MDR-MC and MPPR-MC were coded in FreeBASIC 0.15, an open-source BASIC compiler; this language was chosen for compatibility with

existing code to generate networks as described above. The machine used for these tests was a 3.40 GHz Pentium 4 with 2 GB RAM, running Windows XP.

## 5.4 Computation Time

### *5.4.1 Algorithm MDR-MC*

To investigate the effect of varying problem size on computation time, the magnitude of each of four network parameters (number of nodes, average connectivity [equivalent to number of arcs], number of states of each arc, and maximum arc cost) was adjusted independently of the other three to isolate the impact each has on running time.

The "base network" consists of 50 nodes, an average connectivity of 3, 3 states per arc, and a maximum arc cost of 5. For each of the four experiments, one parameter was increased while the others were held constant. Ten random networks were generated using the specified parameters, and the average running time was determined.

Figures 5.1-5.4 plot the running time as a function of increasing number of nodes, average connectivity, number of states, and maximum arc cost, respectively.

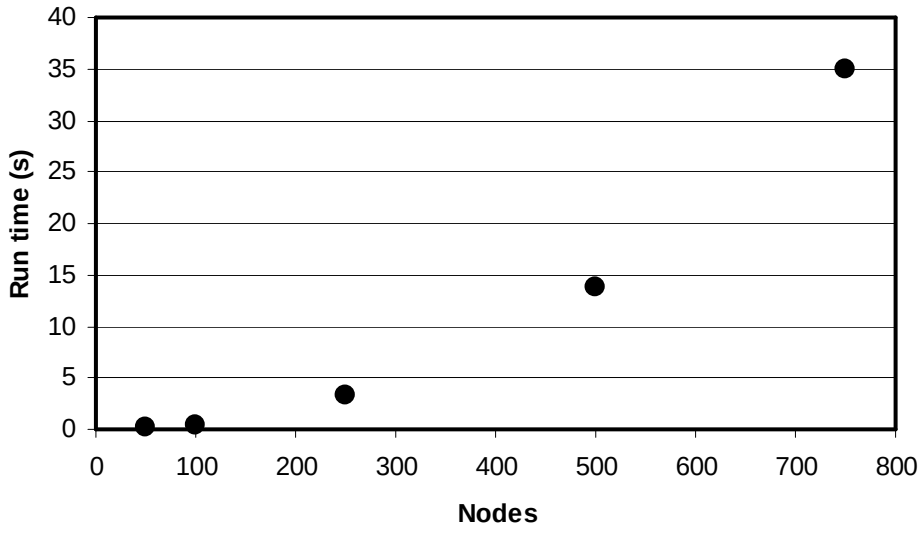


**Figure 5.1.** Computation time of MDR-MC as a function of number of nodes.

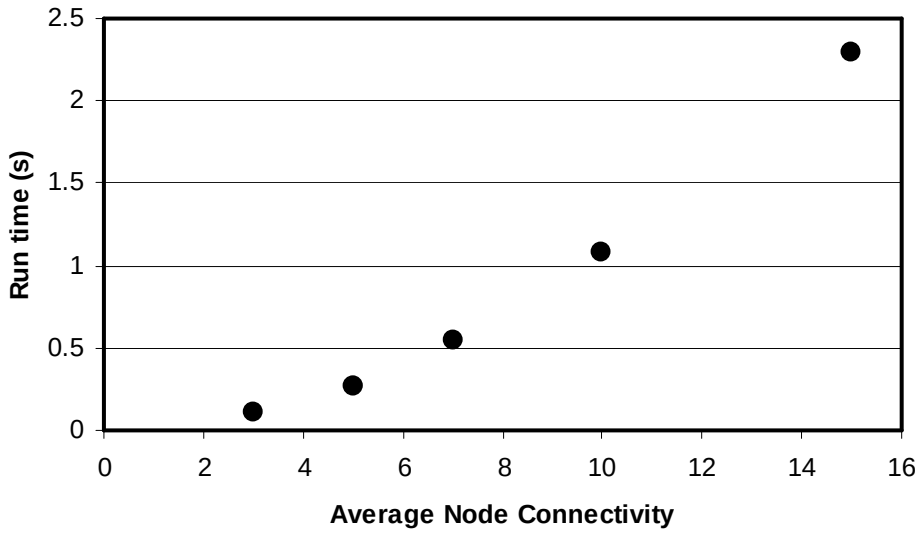


**Figure 5.2.** Computation time of MDR-MC as a function of node connectivity.

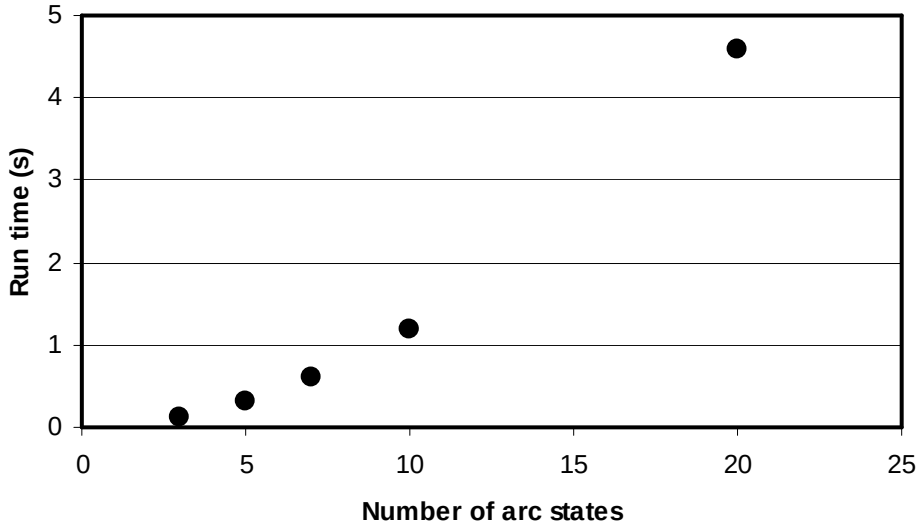


**Figure 5.3.** Computation time of MDR-MC as a function of number of arc states.

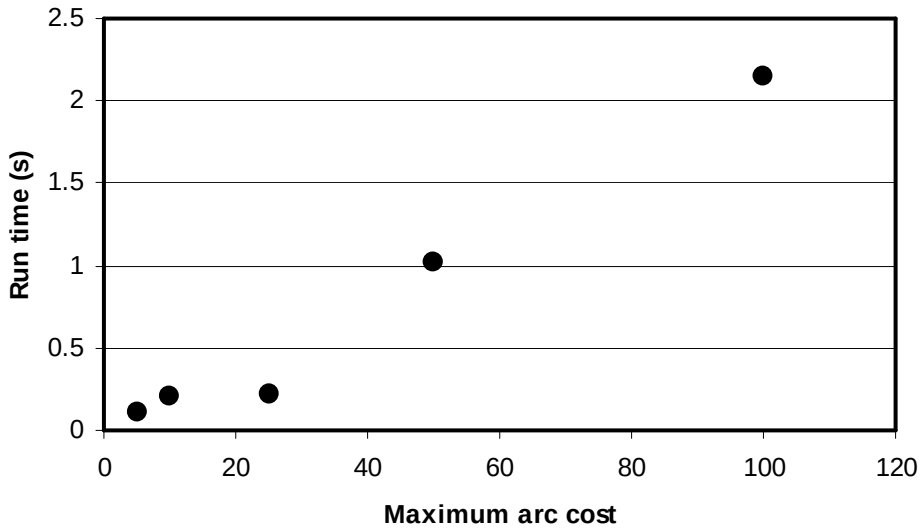


**Figure 5.4.** Computation time of MDR-MC as a function of maximum arc cost.

We see that computation time appears to increase linearly with maximum arc cost, but faster than linearly with respect to the number of nodes, arc connectivity, and number of arc states. This is all in accordance with the $O(Mn^3mS^2)$ complexity result in Proposition 3.3, except for the arc connectivity result. Proposition 3.3 states that in the worst case, computation time grows linearly with $m$; however, this worst-case calculation assumed a complete graph where every arc has $O(n)$ predecessors. For networks that are less dense, the run time will be substantially less than this, but will also depend on the average connectivity. Thus, as $m$ increases in our experiments, and $n$ is held constant, the average connectivity necessarily increases as well, and the faster-than-linear increase in run time can be attributed to a combination of increasing number of arcs and increasing node degree. However, Proposition 3.3 ensures that the growth is no faster than linear once average connectivity is sufficiently high.

An attempt was made to solve a network of 1000 nodes, but the memory requirements exceeded the capacity of the machine used; however, a more efficient implementation might be capable of solving networks of that size.

### *5.4.2 Algorithm MPPR-MC*

Similar to the investigation of Algorithm MDR-MC, the effect of varying problem size on computation time is tested by adjusting the magnitude of six

network parameters. Four are the same as in the previous section: number of nodes, average connectivity (number of arcs), number of arc states, and maximum arc cost. Two additional parameters are varied here: the number of polynomial pieces, and the degree of the polynomials used. The "base case" is the same as that in the previous section, with two polynomial pieces of degree two. As before, the magnitude of each of these six parameters is adjusted independently of the others to isolate the impact of each on computation time.

Figures 5.5-5.11 display these results. Variation of run time with the four parameters common with MDR-MC is similar, and no additional discussion is needed, although it is interesting to note that for both algorithms, no increase in run time is observed when the largest arc cost is increased from 10 to 25. As the number of polynomial pieces increases, nonlinear growth in run time occurs, and as the polynomial degree increases, linear growth is seen in the run time. Both of these results are in accordance with the $O(NMn^3mD^2S^2)$ complexity result in Proposition 4.2.

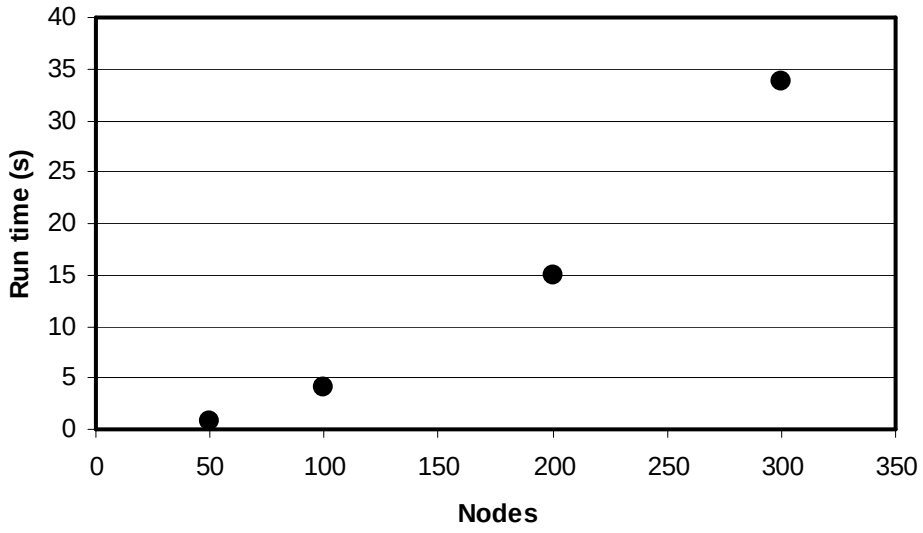


**Figure 5.5.** Computation time of MPPR-MC as a function of number of nodes.

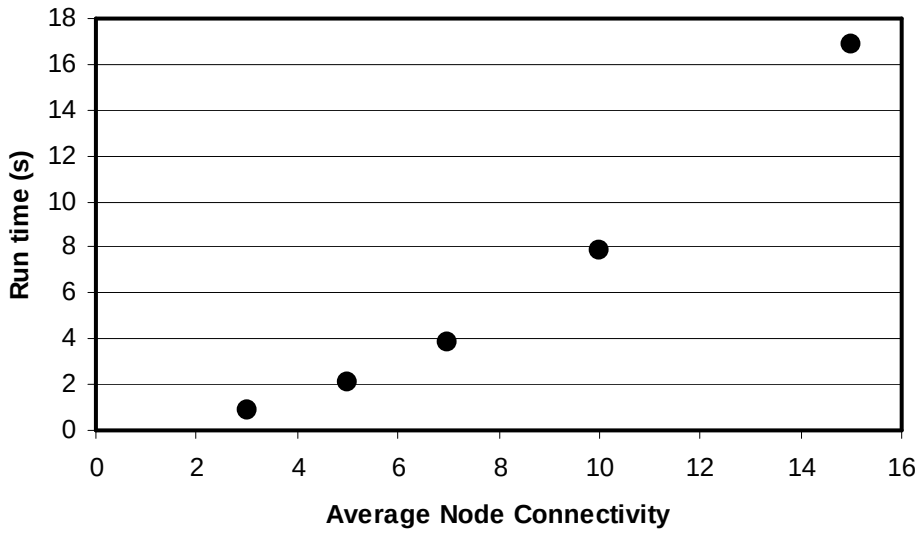


**Figure 5.6.** Computation time of MPPR-MC as a function of node connectivity.

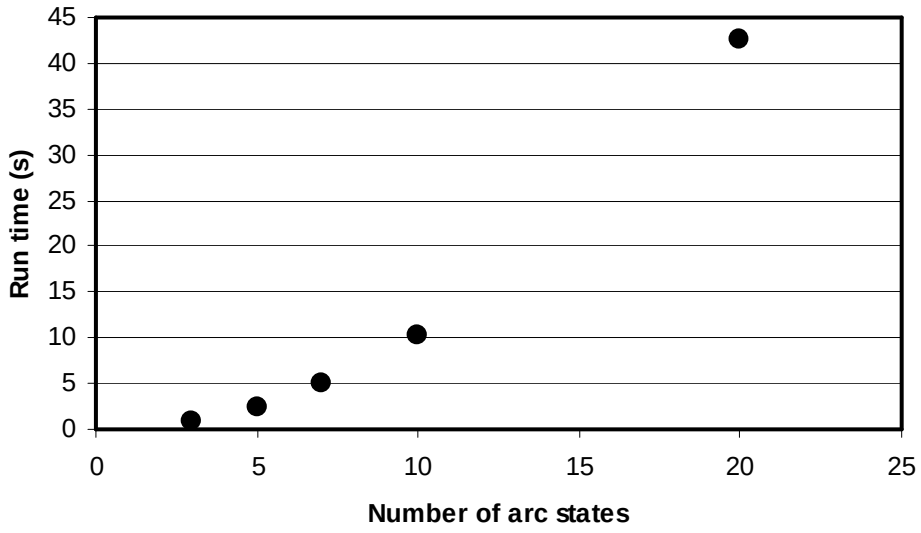


**Figure 5.7.** Computation time of MPPR-MC as a function of number of arc states.

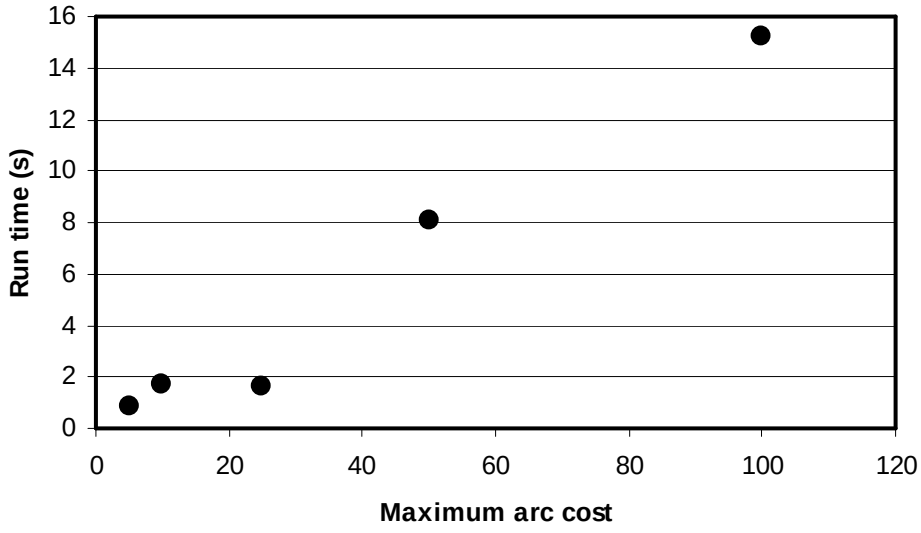


**Figure 5.8.** Computation time of MPPR-MC as a function of maximum arc cost.

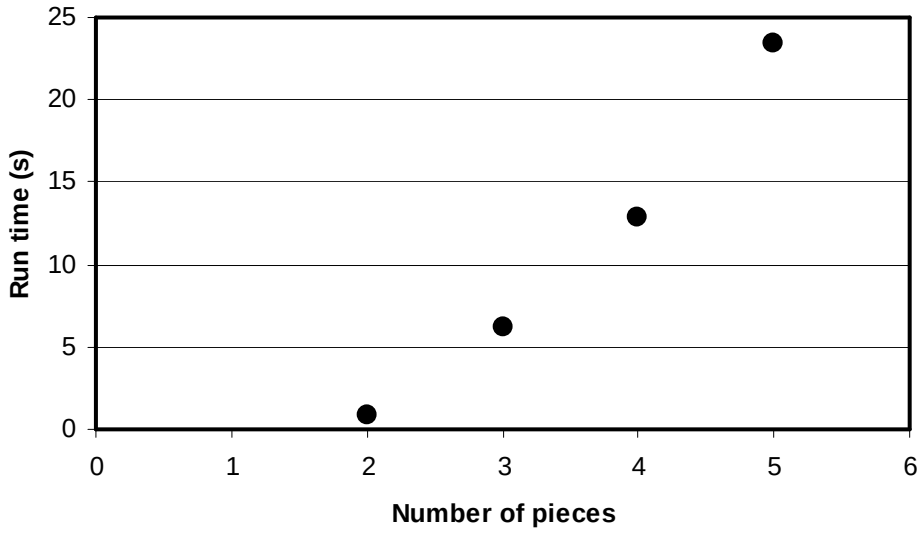


**Figure 5.9.** Computation time of MDR-MC as a function of number of pieces.

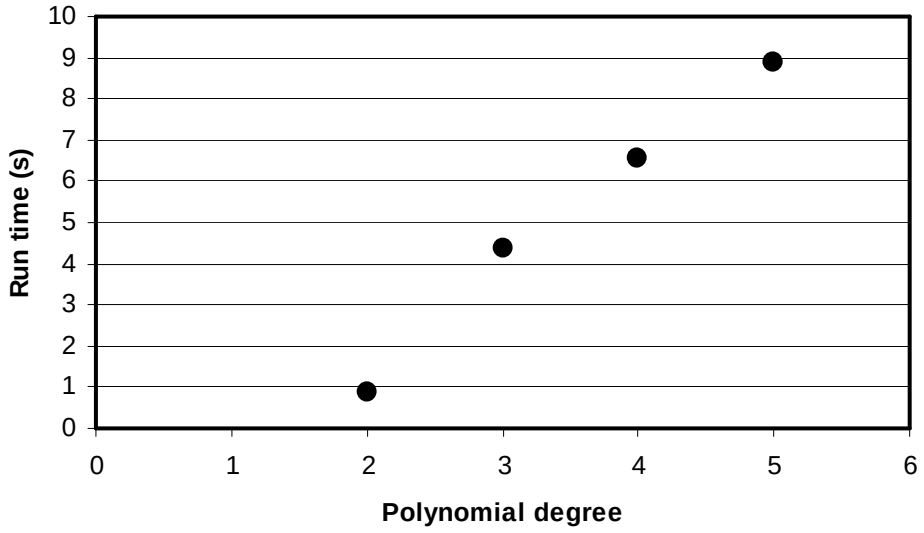


**Figure 5.10.** Computation time of MDR-MC as a function of polynomial degree.

## 5.5 The Reliability/Expected Cost Tradeoff

Using deviance as a disutility function, as in MDR-MC, has a simple interpretation of attempting to arrive at the destination as close as possible to a target arrival time, penalizing the square of any difference. In this section, this objective is compared with that of simply minimizing expected travel time, the online shortest path (OSP) solution. The comparison is most interesting when the target arrival time for MDR-MC is the expected cost of the OSP solution.

From basic optimization theory we know that the MDR solution cannot have expected cost lower than the OSP solution, and likewise, the OSP solution cannot have a lower deviance than the MDR solution. Thus reliability (measured by deviance) is obtained at the expense of the expected cost.

Algorithm MDR-MC was applied to randomly generated networks of various sizes to investigate this tradeoff for each of the three types of probability tensor described in Section 5.2: positive correlation, independence, and negative correlation.

For each probability scenario, networks of 50, 100, 250, and 500 nodes were examined. Fifty random networks of each size were generated, and for each algorithm MDR-MC was applied using the expected cost of the online shortest path on that network as a target. In some cases, the MDR and OSP

policies are identical; Table 5.1 notes the frequency of such occurrences for each combination of network size and correlation structure.

The remaining cases, when the MDR policy is distinct from the OSP policy, are then examined to find the difference in the expected cost, variance, and deviance of the policies. When calculating the deviance of the OSP solution we assume the target is the expected cost; thus, for the OSP solution, deviance and variance are identical. Figures 5.11-5.13 plot the average percent difference, as measured from the OSP solution; numerical values can be found in Tables 5.2-5.4.

**Table 5.1**. Frequency of identical solutions between OSP and MDR-MC.

| Nodes | Positive | Independent | Negative |
|---|---|---|---|
| 50 | 66% | 52% | 62% |
| 100 | 56% | 54% | 54% |
| 250 | 40% | 48% | 48% |
| 500 | 36% | 58% | 48% |

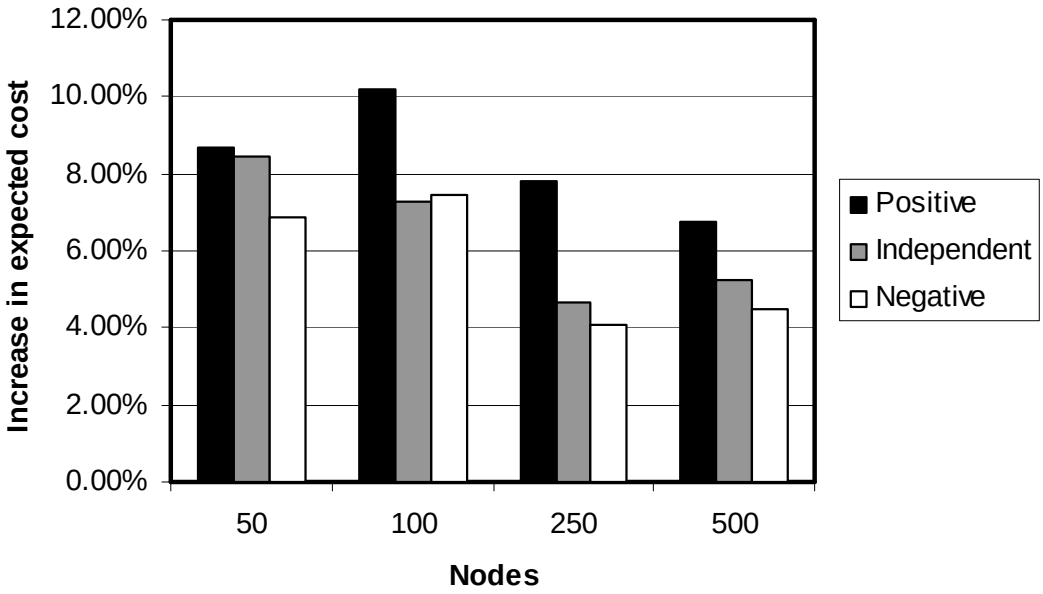


**Figure 5.11**. Increase in expected cost of MDR solution.

**Table 5.2**. Increase in expected cost of MDR solution.

| Nodes | Positive | Independent | Negative |
|---|---|---|---|
| 50 | 8.71% | 8.47% | 6.86% |
| 100 | 10.20% | 7.27% | 7.44% |
| 250 | 7.79% | 4.68% | 4.08% |
| 500 | 6.75% | 5.24% | 4.50% |

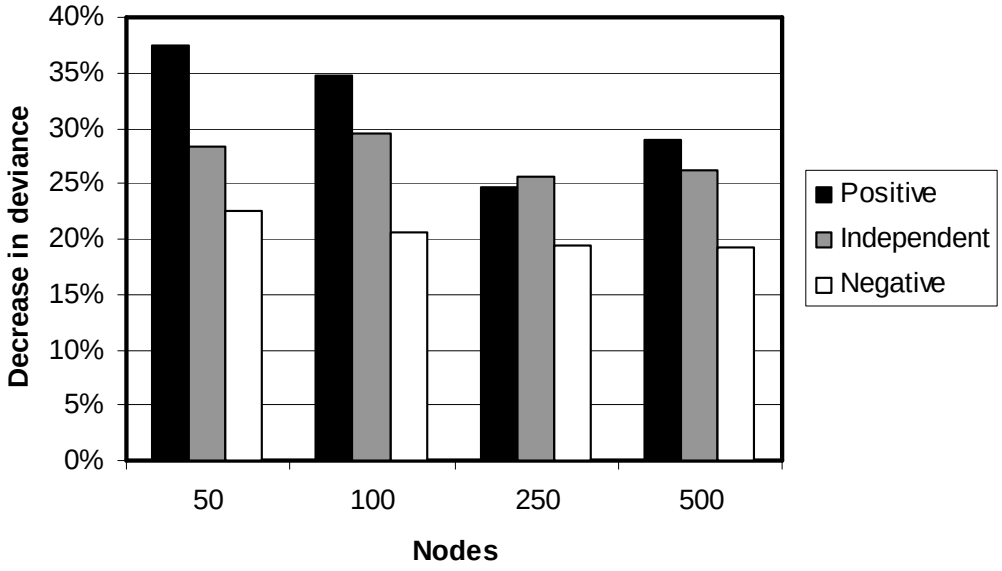


**Figure 5.12**. Deviance reduction for MDR solution.

**Table 5.3**. Deviance reduction for MDR solution.

| Nodes | Positive | Independent | Negative |
|---|---|---|---|
| 50 | 37.40% | 28.42% | 22.44% |
| 100 | 34.67% | 29.48% | 20.58% |
| 250 | 24.73% | 25.55% | 19.35% |
| 500 | 28.93% | 26.15% | 19.21% |

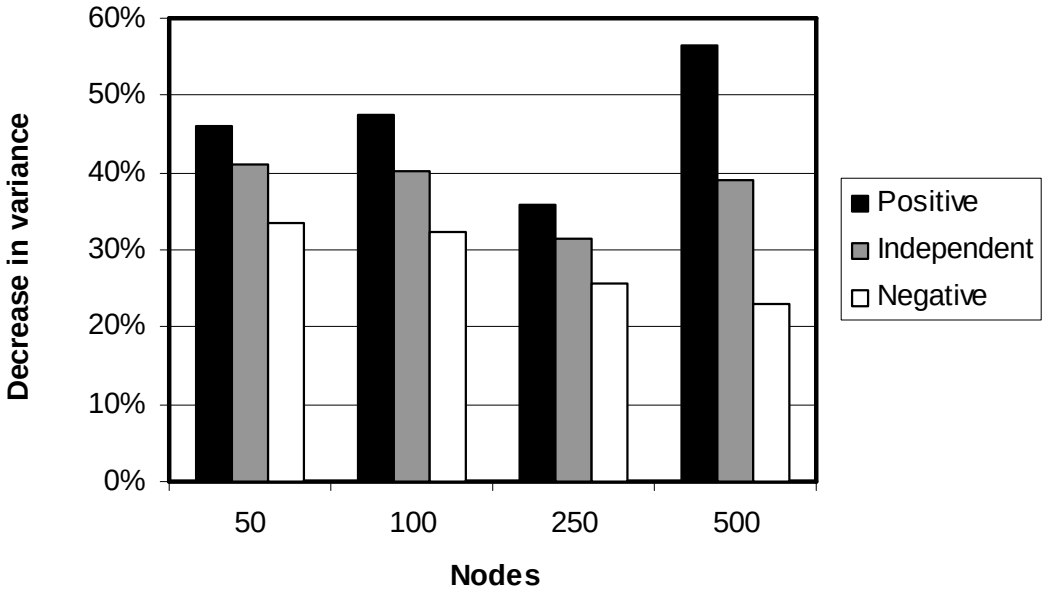


**Figure 5.13**. Reduction in variance for MDR solution.

**Table 5.4.** Reduction in variance for MDR solution.

| Nodes | Positive | Independent | Negative |
|---|---|---|---|
| 50 | 46.13% | 41.12% | 33.59% |
| 100 | 47.36% | 40.18% | 32.37% |
| 250 | 35.71% | 31.55% | 25.77% |
| 500 | 56.54% | 38.88% | 22.94% |

The OSP and MDR solutions are identical for about half of the networks tested; but when they differ, the relative increase in mean cost is small compared to the reduction in deviance and variance. For the cases where arc costs are correlated, the frequency of identical solutions decreases with network size. This makes intuitive sense, as large networks offer more potential policies for producing a lower deviance than exhibited in the OSP solution.

Further, it is seen that the improvement in deviance and variance is greatest when arc costs are positive correlated, although the networks in this case also exhibit the greatest increase in expected cost as well. Again, this is in accordance with intuition, as more information about the total path cost can be inferred with this type of dependency. When arc costs have negative correlation, attempts to influence the travel cost by choosing an arc with high or low cost face "resistance" in the dependency structure, as the following arc is more likely to exist in a state that counteracts the immediate decision. Nevertheless, the observation that the algorithm reduces deviance and variance the most when arc costs are positively correlated is favorable, since one expects traffic networks to exhibit this type of dependency structure.

## 5.6 The Value of Online Solution

Additionally, we seek to quantify the benefit of allowing an adaptive policy compared to an offline one where the entire path must be specified *a priori*. Finding an *a priori* minimum deviance (or minimum variance) path is considerably more difficult, and can be viewed as a constrained version of the online problem. Although not studied extensively, we speculate that this problem is NP-complete.

Thus, for the purposes of this numerical comparison, the minimum deviance paths were found through complete enumeration for the 50-node networks; attempting to enumerate all paths in the larger networks requires excessive computation time. Table 5.5 compares the average deviance of the optimal online and *a priori* solutions. The deviance of the optimal MDR policy is always lower than that of the *a priori* solution, since it is a less constrained version of the same problem.

**Table 5.5**. Optimal deviance values for *a priori* and online solutions.

| Solution | Positive | Independent | Negative |
|---|---|---|---|
| *a priori* | 16.1 | 14.2 | 9.83 |
| recourse | 15.4 | 12.5 | 9.23 |

## 5.7 Conclusions

In this section a method for network generation was presented, followed by results from implementation of algorithms MDR-MC and MPPR-MC. An analysis of computation time as problem size grew was mostly consistent with the complexity results derived in Chapters 3 and 4; the only discrepancy regarded the increase in computation time with number of arcs, but this is found to occur because the network size is far smaller than the worst case of a complete graph.

Additionally, when examining MDR-MC, it is found that substantial decrease in solution variance can be obtained for a relatively small cost in expected travel time. This indicates that correctly accounting for a user's preferences regarding reliability is of critical importance for modeling the route choice decision, as travelers may not follow the least expected cost policy if they are even slightly risk averse.

Further, this improvement in variance was found to be largest in networks which exhibit positive correlation. This is encouraging, as this correlation structure is most likely to represent congested networks.

# Chapter 6: Conclusions

## 6.1 Implications of Work

It is becoming clear that accounting for uncertainty in transportation networks is a crucial step towards improved, more realistic models. The work described here provides an individual routing policy model which accounts for operational uncertainty in two ways: by accounting for a user's attitude towards risk regarding travel times, and by accounting for a user's ability to respond *en route* to information received about uncertain network conditions.

This thesis presents the first algorithm which finds an optimal routing policy in a stochastic, time-dependent network where the traveler's preferences are determined by an arbitrary piecewise polynomial disutility function. Such utility functions are applicable in cases where one's preferences are highly nonlinear, as in time-sensitive supply chains, or when an individual is driving to an important meeting. While a variety of models have been developed to represent this behavior, the methods presented here are more general.

By accounting for recourse behavior, one can also begin to examine the impact of information provision on route choice in the context of nonlinear preferences. Although the model developed here is an extremely simplified

version of reality which accounts only for limited dependencies, this work represents a first step towards a more sophisticated model incorporating more realistic dependencies and information provision strategies. Such a model could contribute to the evaluation of ITS policies that provide information to drivers.

For one particular type of disutility function, deviance, numerical results showed that substantial reductions in variance can be attained for a relatively small increase in expected travel time. Thus, the assumption that travelers value reliability much less than expected cost (or not at all) carries the risk of mis-specifying the paths of many users of the network.

## 6.2 Future Work

Many opportunities exist to further this research into modeling the effect of uncertainty on traveler behavior. Figure 6.1 graphically illustrates the relationship between this thesis, indicated in the shaded box, and other components of a more comprehensive model of operational uncertainty in transportation networks.

.

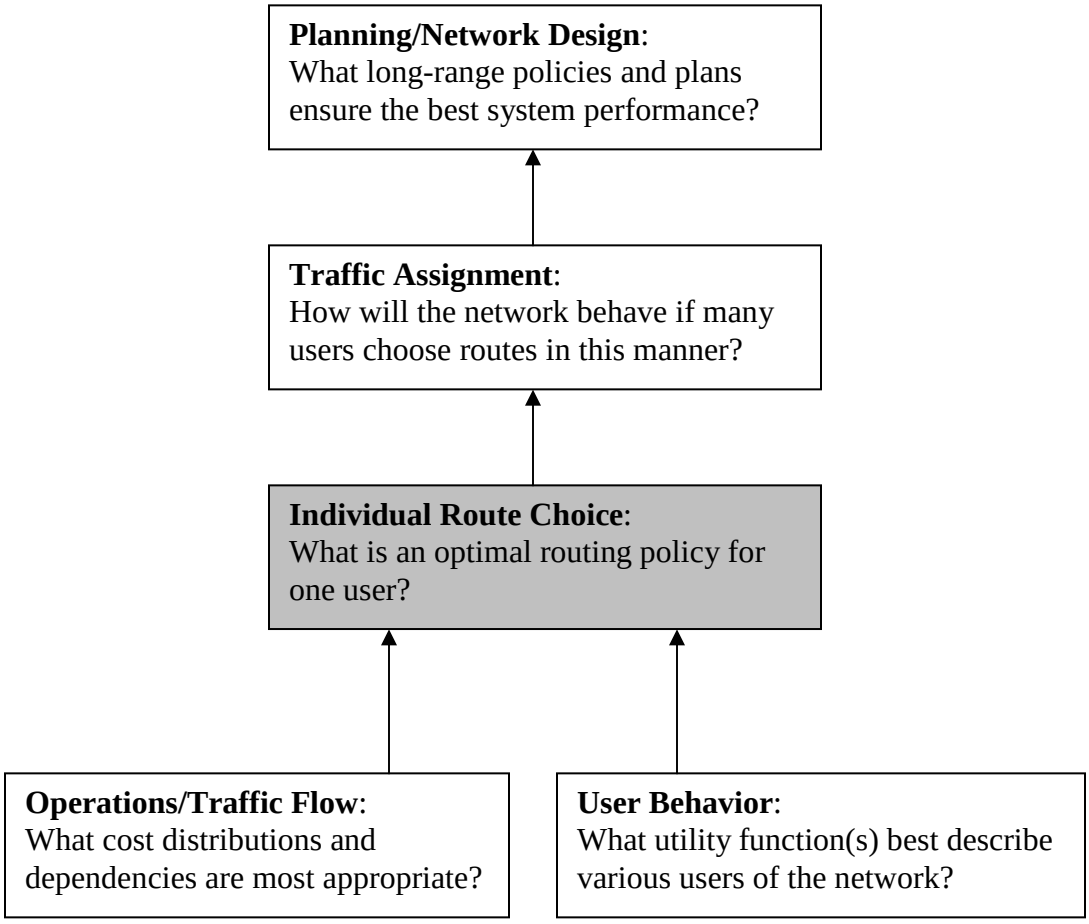


**Figure 6.1**. A comprehensive model of operational uncertainty in transportation planning.

To fully model the impacts of uncertainty, contributions need to be made in each of the other areas indicated in Figure 6.1. Ultimately, we seek to know how uncertainty impacts long-term transportation planning, and how policies should be best selected given that this uncertainty exists

However, to evaluate any plan or policy, it is necessary to know how users of the network will respond, illustrated in the box labeled "Traffic Assignment." Since users typically behave in a selfish manner, it is difficult to predict the impacts of these policies, and the outcome may be counterintuitive. Thus, before one can develop a methodology for incorporating uncertainty into transportation planning, one must be able to model how a large number of users, with a variety of preferences regarding risk, and a variety of access to information, will respond to any changes to the network. This problem is further compounded because each user's choice, which depends on the cost and level of congestion on links in the network, is itself dependent on the choice of all other users.

Still, a first step towards this problem is to develop a model for how a single user should choose a routing policy, given that the decisions of all others are fixed. The work in this thesis is directed towards this step.

However, the algorithms developed in this thesis in turn rely upon correct parameters regarding the link travel time distributions, the nature of the spatial and temporal dependencies, the information provided to users, and correct specification of a user's utility function.

Thus, much work still remains in developing a full treatment of operational uncertainty in traffic networks, and there are many opportunities for valuable contributions to be made. Amidst all the randomness and variation that exists in transportation networks, one statement can be made with absolute certainty: accounting for the impacts of uncertainty is of enormous importance to transportation modeling and planning. While much research is required to fully realize such a model, this thesis nevertheless provides an important step in this direction.

# Appendix A: Example of MDR-MC

This appendix illustrates how algorithm MDR-MC finds a minimum deviance policy on a simple network, by showing the results of each step of the algorithm until termination. The network used is shown below, in Figure A.1. Table A.1 describes the costs of the arcs, and shows the state indices assigned to each cost. Arcs 1 and 4 have deterministic cost, while arcs 2, 3, and 5 are stochastic. Arc 3 exhibits temporal dependency, while Arc 5 exhibits spatial dependency. Table A.2 lists all of the NTPCs corresponding to this network (the set $\Phi$), and Table A.3 lists all realizations of the network (the set $\Omega$), which correspond to all possible paths from the origin *O* to the destination *D*. In this example the target arrival time is 4; Figure A.4 plots the disutility of arrival for each possible arrival time.

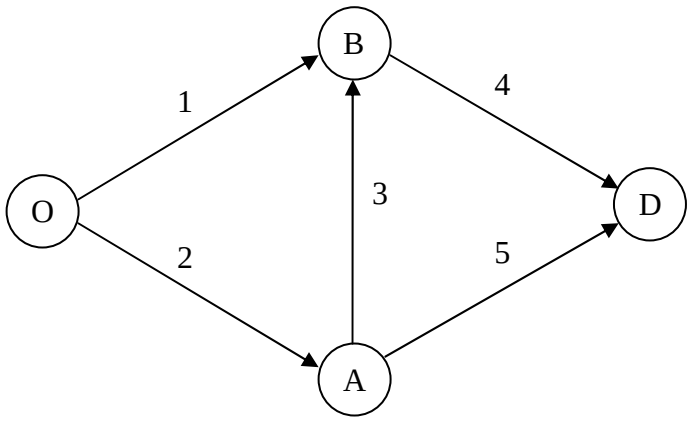


**Figure A.1.** Sample network to illustrate algorithm MDR-MC.

**Table A.1.** Arc costs for the sample network.

| Arc | Cost | State |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 2 with probability 0.5 | 1 |
| | 3 with probability 0.5 | 2 |
| 3 | 1, if departing at time 2 | 1 |
| | 2, if departing at time 3 | 2 |
| 4 | 1 | 1 |
| 5 | *If arc 2 had cost 2*: | |
| | 1, with probability 0.8 | 1 |
| | 2, with probability 0.2 | 2 |
| | *If arc 2 had cost 3*: | |
| | 1, with probability 0.2 | 1 |
| | 2, with probability 0.8 | 2 |

**Table A.2.** List of NTPCs for the sample network (the set $\Phi$).

| NTPC* |
|---|
| (O,0,∅,0) |
| (A,2,2,1) |
| (A,3,2,2) |
| (B,2,1,1) |
| (B,3,3,1) |
| (B,5,3,2) |
| (D,3,4,1) |
| (D,3,5,1) |
| (D,4,4,1) |
| (D,4,5,1) |
| (D,4,5,2) |
| (D,5,5,2) |
| (D,6,4,1) |

*Each NTPC is a quadruple (Node, Time, Predecessor Arc, Predecessor State)

**Table A.3.** List of realizations of the sample network (the set Ω).

| Realization* | Cost |
|---|---|
| (O,0,∅,0), (B,2,1,1), (D,3,4,1) | 3 |
| (O,0,∅,0), (A,2,2,1), (B,3,3,1), (D,4,4,1) | 4 |
| (O,0,∅,0), (A,2,2,1), (D,3,4,1) | 3 |
| (O,0,∅,0), (A,2,2,1), (D,4,4,2) | 4 |
| (O,0,∅,0), (A,3,2,2), (B,5,3,1), (D,6,4,1) | 6 |
| (O,0,∅,0), (A,3,2,2), (D,4,4,1) | 4 |
| (O,0,∅,0), (A,3,2,2), (D,5,4,2) | 5 |

*Each realization is a subset of Φ that represents a path from *O* to *D*

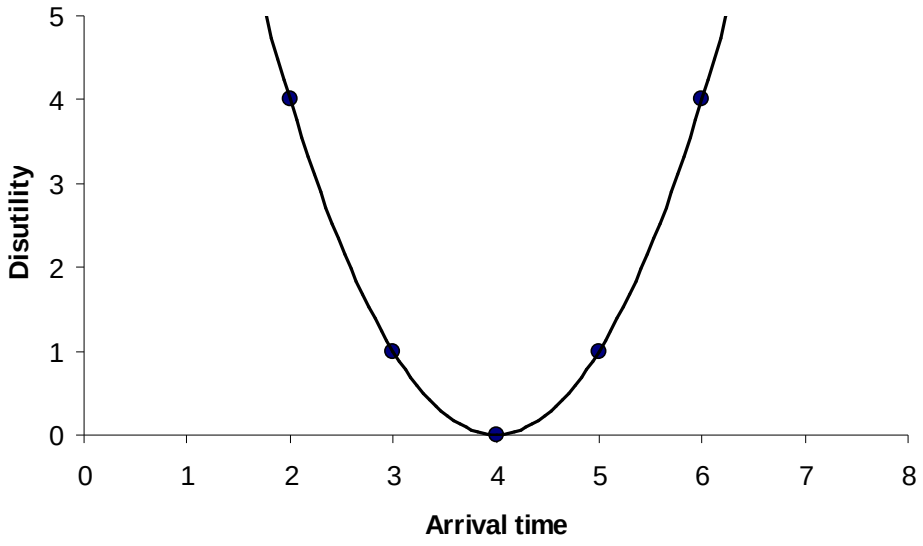


**Figure A.2.** Deviance as a disutility function (target arrival time is 4)

We now explain each step of the algorithm in detail. The reader may find it useful to refer to Table A.4, which lists the labels and scan eligible list at each stage of the algorithm.

**Initialization**: For all destination NTPCs, we set the $E$ and $V$ labels to zero, and the $D$ labels to the deviance corresponding to the arrival time, as shown in Figure A.2. $E$, $V$, and $D$ labels at all other NTPCs are set to infinity. SEL is initialized to contain all node-time pairs that can reach the destination by traversing only one arc.

**Iteration 1**: Remove node-time pair (A,2) from SEL. Since the only way to reach (A,2) is via arc 2 in state 1, in the *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* Calculate the expected cost, variance, and deviance from following this arc ($E_3$, $V_3$, and $D_3$) using the formulas in Corollary 3.2. Since the downstream labels are all infinity, so are $E_3$, $V_3$, and $D_3$. This is no better than the current labels at (A,2), so no labels are updated and no node-time pairs are added to SEL.

*Arc 5*: Using Corollary 3.2, we find $E_5$, $V_5$, and $D_5$ using the labels at the downstream NTPCs:

$E_5$(A,2,2,1) = 0.8(1 + 0) + 0.2(2 + 0) = 1.2

$V_5(A,2,2,1) = 0.8((0 + 1 - 1.2)^2 + 0) + 0.2((0 + 2 - 1.2)^2 + 0) = 0.16$

$D_5(A,2,2,1) = 0.16 + (1.2 - (4 - 2))^2 = 0.8$

Since $0.8 < \infty$ we set the labels at (A,2,2,1) to these temporary values, and set the path pointer $\pi$ to arc 5. Node-time pair (O,0) is the only predecessor of (A,2), and is added to SEL.

**Iteration 2**: Remove node-time pair (B,3) from SEL. Since the only way to reach (B,3) is via arc 3 in state 1, and the only outgoing arc is arc 4, the *for* loop consists only of a single evaluation:

*Arc 4:* Using Corollary 3.2, we find $E_4$, $V_4$, and $D_4$ using the labels at the downstream NTPCs:

$E_4(B,3,3,1) = 1(1 + 0) = 1$

$V_4(B,3,3,1) = 1((0 + 1 - 1)^2 + 0) = 0$

$D_4(B,3,3,1) = 0 + (1 - (4 - 3))^2 = 0$

Since $0 < \infty$ we set the labels at (B,3,3,1) to these temporary values, and set the path pointer $\pi$ to arc 4. Node-time pair (O,0) is a predecessor of (B,3), but is already in SEL; node-time pair (A,2) is a predecessor of (B,3) that has already been scanned, but is no longer in SEL. Therefore, we must add (A,2) to SEL.

**Iteration 3**: Remove node-time pair (A,2) from SEL. (This is the second time it has been scanned.) As before, in the *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* Proceeding in the same manner, we find

$E_3(\text{A},2,2,1) = 1(1 + 1) = 2$

$V_3(\text{A},2,2,1) = 1((1 + 1 - 2)^2 + 0) = 0$

$D_3(\text{A},2,2,1) = 0 + (2 - (4 - 2))^2 = 0$

Since $0 < 0.8$ we set the labels at (A,2,2,1) to these temporary values, and update the path pointer $\pi$ to arc 3. That is, after step 1, arc 5 was the preferred arc when arriving at node A at time 2, but in this step, choosing arc 3 was seen to produce a lower deviance, so we update the policy accordingly. Node-time pair (O,0) is the only predecessor of (A,2), and is already in SEL.

*Arc 5*: As before,

$E_5(\text{A},2,2,1) = 0.8(1 + 0) + 0.2(2 + 0) = 1.2$

$V_5(\text{A},2,2,1) = 0.8((0 + 1 - 1.2)^2 + 0) + 0.2((0 + 2 - 1.2)^2 + 0) = 0.16$

$D_5(\text{A},2,2,1) = 0.16 + (1.2 - (4 - 2))^2 = 0.8$

Since $0.8 > 0$ the labels are not updated, and no node-time pairs are considered for addition to SEL.

**Iteration 4**: Remove node-time pair (A,3) from SEL. As the only possible combination of predecessor arc and predecessor state is arc 2 in state 2, in the *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* The downstream labels are all infinity, so following arc 3 will not improve the routing policy. No labels are updated and SEL is not changed.

*Arc 5*:

$E_5(A,3,2,2) = 0.2(1 + 0) + 0.8(2 + 0) = 1.8$

$V_5(A,3,2,2) = 0.2((0 + 1 - 1.8)^2 + 0) + 0.8((0 + 2 - 1.8)^2 + 0) = 0.16$

$D_5(A,3,2,2) = 0.16 + (1.8 - (4 - 3))^2 = 0.8$

Since $0.8 < \infty$ the labels are updated, and the path pointer is set to arc 5. The only predecessor is (O,0) which is already in SEL.

**Iteration 5**: Remove node-time pair (B,5) from SEL. As in Iteration 2, the *for* loop consists of a single evaluation:

$E_4(B,5,3,2) = 1(1 + 0) = 1$

$V_4(B,5,3,2) = 1((0 + 1 - 1)^2 + 0) = 0$

$D_4(B,5,3,2) = 0 + (1 - (4 - 5))^2 = 4$

Since $4 < \infty$ the labels are updated, and the path pointer is set to arc 4. Predecessor (O,0) is already in SEL; predecessor (A,3) is added to SEL for the second time.

**Iteration 6**: Remove node-time pair (A,3) from SEL. As in Iteration 4, in the *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:*

$E_3(A,3,2,2) = 1(2 + 1) = 3$

$V_3(A,3,2,2) = 1((1 + 2 - 3)^2 + 0) = 0$

$D_3(A,3,2,2) = 0 + (3 - (4 - 3))^2 = 4$

The current deviance label at (A,3,2,2) is 1.8. Since 4 > 1.8, no updating occurs. That is, arc 5 remains the superior choice at this NTPC.

*Arc 5*: These temporary labels are found to be the same as before, so again, no updating occurs.

**Iteration 7**: Remove (B,2) from SEL. As in Iteration 2, the *for* loop consists of a single evaluation:

$E_4(B,2,1,1) = 1(1 + 0) = 1$

$V_4(B,2,1,1) = 1((0 + 1 - 1)^2 + 0) = 0$

$D_4(B,2,1,1) = 0 + (1 - (4 - 2))^2 = 1$

Since $1 < \infty$ the labels are updated, and the path pointer is set to arc 4. The only predecessor, (O,0), is already in SEL.

**Iteration 8**: Remove (O,0) from SEL. We calculate the labels as usual for the two choices of outgoing arc:

*Arc 1:*

$E_1(O,0,\varnothing,0) = 1(1 + 2) = 3$

$V_1(O,0,\varnothing,0) = 1((1 + 2 - 3)^2 + 0) = 0$

$D_1(O,0,\varnothing,0) = 0 + (3 - (4 - 0))^2 = 1$

Since $1 < \infty$ we update labels at the origin and set the path pointer to arc 1.

*Arc 2:*

$E_2(O,0,\varnothing,0) = 0.5(2 + 2) + 0.5(3 + 1.8) = 4.4$

$V_2(O,0,\varnothing,0) = 0.5((2 + 2 - 4.4)^2 + 0) + 0.5((1.8 + 3 - 4.4)^2 + 0.16) = 0.24$

$D_2(O,0,\varnothing,0) = 0.24 + (4.4 - (4 - 0))^2 = 0.4$

Since $0.4 < 1$ we update the labels at the origin and change the path pointer to arc 2. At this point SEL is empty, and we terminate the algorithm.

Table A.4 shows the labels and contents of SEL at each step of the algorithm, as discussed in detail above. Table A.5 displays the final routing policy, by showing the choice of outgoing node at each NTPC (excluding the destination NTPCs). Table A.6 lists each realization of the network, along with the probability that it occurs when following the policy in Table A.5. From Table A.6 one may readily verify that the labels at the origin do indeed correspond to the mean, variance, and deviance of travel time.

**Table A.4.** Labels and SEL at each iteration of the algorithm.

| Step: | Initialize | | | | 1 | | | | 2 | | | | 3 | | | | 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scan: | N/A | | | | (A,2) | | | | (B,3) | | | | (A,2) | | | | (A,3) | | | |
| NTPC | E | V | D | $\pi$ | E | V | D | $\pi$ | E | V | D | $\pi$ | E | V | D | $\pi$ | E | V | D | $\pi$ |
| (O,0,∅,0) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - |
| (A,2,2,1) | ∞ | ∞ | ∞ | – | 1.2 | 0.16 | 0.8 | 5 | 1.2 | 0.16 | 0.8 | 5 | 2 | 0 | 0 | 3 | 2 | 0 | 0 | 3 |
| (A,3,2,2) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | 1.8 | 0.16 | 0.8 | 5 |
| (B,2,1,1) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - |
| (B,3,3,1) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | 1 | 0 | 0 | 4 | 1 | 0 | 0 | 4 | 1 | 0 | 0 | 4 |
| (B,5,3,2) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - |
| (D,3,4,1) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,3,5,1) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,4,4,1) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,4,5,1) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,4,5,2) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,5,5,2) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,6,4,1) | 0 | 0 | 4 | – | 0 | 0 | 4 | - | 0 | 0 | 4 | - | 0 | 0 | 4 | - | 0 | 0 | 4 | - |
| SEL: | (B,2)<br>(B,3)<br>(B,5)<br>(A,2)<br>(A,3) | | | | (B,2)<br>(B,3)<br>(B,5)<br>(A,2)<br>(O,0) | | | | (B,2)<br>(B,5)<br>(A,3)<br>(O,0)<br>(A,2) | | | | (B,2)<br>(B,5)<br>(A,3)<br>(O,0) | | | | (B,2)<br>(B,5)<br>(O,0) | | | |

| Step: | 5 | | | | 6 | | | | 7 | | | | 8 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scan: | (B,5) | | | | (A,3) | | | | (B,2) | | | | (O,0) | | | |
| NTPC | E | V | D | $\pi$ | E | V | D | $\pi$ | E | V | D | $\pi$ | E | V | D | $\pi$ |
| (O,0,∅,0) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | - | ∞ | ∞ | ∞ | - | 4.4 | 0.24 | 0.4 | 2 |
| (A,2,2,1) | 2 | 0 | 0 | 3 | 2 | 0 | 0 | 3 | 2 | 0 | 0 | 3 | 2 | 0 | 0 | 3 |
| (A,3,2,2) | 1.8 | 0.16 | 0.8 | 5 | 1.8 | 0.16 | 0.8 | 5 | 1.8 | 0.16 | 0.8 | 5 | 1.8 | 0.16 | 0.8 | 5 |
| (B,2,1,1) | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | – | 1 | 0 | 1 | 4 | 1 | 0 | 1 | 4 |
| (B,3,3,1) | 1 | 0 | 0 | 4 | 1 | 0 | 0 | 4 | 1 | 0 | 0 | 4 | 1 | 0 | 0 | 4 |
| (B,5,3,2) | 1 | 0 | 4 | 4 | 1 | 0 | 4 | 4 | 1 | 0 | 4 | 4 | 1 | 0 | 4 | 4 |
| (D,3,4,1) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,3,5,1) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,4,4,1) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,4,5,1) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,4,5,2) | 0 | 0 | 0 | – | 0 | 0 | 0 | - | 0 | 0 | 0 | - | 0 | 0 | 0 | - |
| (D,5,5,2) | 0 | 0 | 1 | – | 0 | 0 | 1 | - | 0 | 0 | 1 | - | 0 | 0 | 1 | - |
| (D,6,4,1) | 0 | 0 | 4 | – | 0 | 0 | 4 | - | 0 | 0 | 4 | - | 0 | 0 | 4 | - |
| SEL: | (B,2)<br>(O,0)<br>(A,3) | | | | (B,2)<br>(O,0) | | | | (O,0) | | | | empty | | | |

**Table A.5.** Minimum deviance policy for this network.

| NTPC | Choice of arc |
|---|---|
| (O,0,∅,0) | 2 |
| (A,2,2,1) | 3 |
| (A,3,2,2) | 5 |
| (B,2,1,1) | 4 |
| (B,3,3,1) | 4 |
| (B,5,3,2) | 4 |

**Table A.6.** Probability of each realization occuring under this policy.

| Realization | Cost | Probability |
|---|---|---|
| (O,0,∅,0), (B,2,1,1), (D,3,4,1) | 2 | 0 |
| (O,0,∅,0), (A,2,2,1), (B,3,3,1), (D,4,4,1) | 4 | 0.5 |
| (O,0,∅,0), (A,2,2,1), (D,3,4,1) | 3 | 0 |
| (O,0,∅,0), (A,2,2,1), (D,4,4,2) | 4 | 0 |
| (O,0,∅,0), (A,3,2,2), (B,5,3,1), (D,6,4,1) | 6 | 0 |
| (O,0,∅,0), (A,3,2,2), (D,4,4,1) | 4 | 0.1 |
| (O,0,∅,0), (A,3,2,2), (D,5,4,2) | 5 | 0.4 |

# Appendix B: Example of MPPR-MC

This appendix illustrates how algorithm MPPR-MC finds a minimum deviance policy on a simple network, by showing the results of each step of the algorithm until termination. The network used is the same as that in Appendix A; for convenience, the network and cost data are reproduced below, in Figure B.1 and Table B.1. The sets $\Phi$ and $\Omega$ are the same as shown in Table A.2 and A.3.

However, here the disutility function is piecewise polynomial; namely,

$$P(t) = \begin{cases} 4 - t & t \in [0, 4] \\ 3(t-4)^2 & t \in (4, \infty) \end{cases} \tag{B.1}$$

The possible arrival times in this network are 3, 4, 5, and 6, and the disutilities associated with arriving at each of these times are 1, 0, 3, and 12, respectively, as shown in Figure B.2. One situation where such a disutility function might apply is that of a person trying to arrive at the airport to catch a flight. Arrival at $t = 4$ is ideal: there is ample time before the flight, but the wait is not excessive. Arriving at $t = 5$, there is still time to catch the flight, but just barely so, and our traveler has to rush to make it on time, and worries about long security lines. By $t = 6$, the flight has already left, causing a large amount of distress to the traveler who must make new travel arrangements. Conversely, arriving at $t = 3$ is relatively stress-free, but the wait is long enough that the traveler feels that time is being

wasted. Naturally, the discomfort from an early arrival is somewhat less than the stress caused by a late arrival.

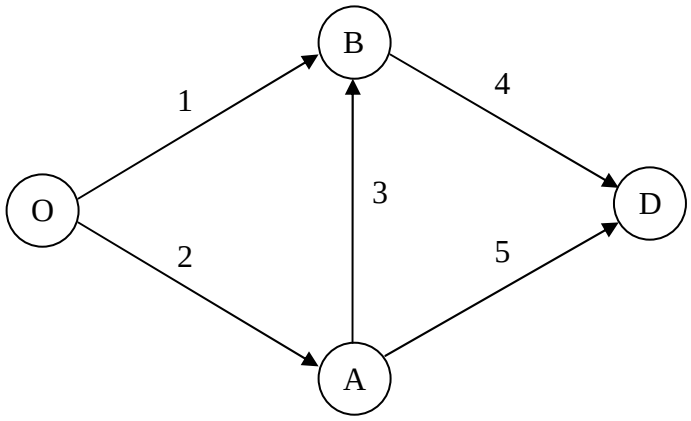


**Figure B.1.** Sample network to illustrate algorithm MPPR-MC.

**Table B.1.** Arc costs for the sample network.

| Arc | Cost | State |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 2 with probability 0.5 | 1 |
| | 3 with probability 0.5 | 2 |
| 3 | 1, if departing at time 2 | 1 |
| | 2, if departing at time 3 | 2 |
| 4 | 1 | 1 |
| 5 | *If arc 2 had cost 2*: | |
| | 1, with probability 0.8 | 1 |
| | 2, with probability 0.2 | 2 |
| | *If arc 2 had cost 3*: | |
| | 1, with probability 0.2 | 1 |
| | 2, with probability 0.8 | 2 |

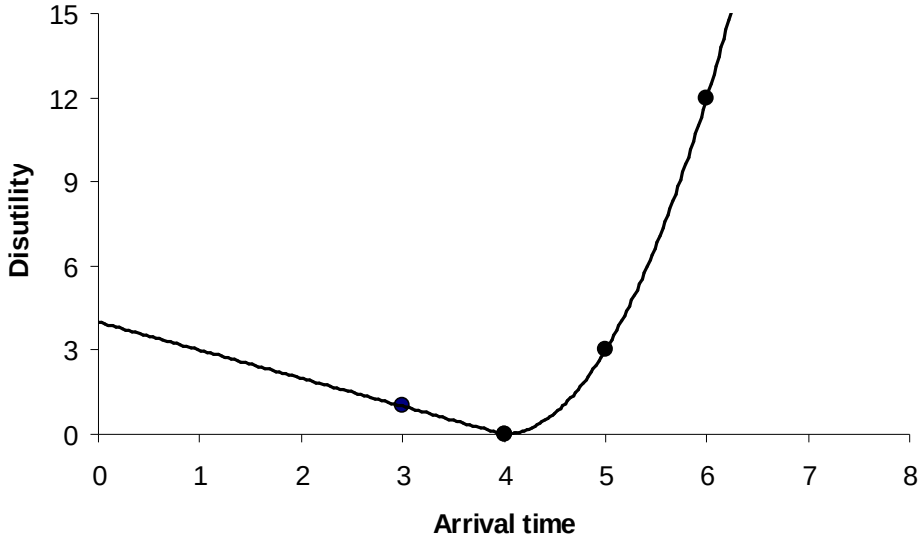


**Figure B.2.** Piecewise polynomial disutility function for this example.

We now explain each step of the algorithm in detail. The reader may find it useful to refer to Table B.2, which lists the labels and scan eligible list at each stage of the algorithm. The labels $\rho_1$ and $\rho_2$ correspond to the probability of arriving in the intervals [0, 4] and (4, $\infty$), respectively; and the labels of the form $L_{ij}$ denote the $j$-th moment of remaining travel time, conditional on the event that arrival occurs during interval $i$. $P$ denotes the expected disutility, and, as before, $\pi$ is the path pointer. With our disutility function, the coefficient vectors are $\mathbf{a}_1 = [4 \;\; -1]$ and $\mathbf{a}_2 = [48 \;\; -24 \;\; 3]$, as can be verified by expanding (B.1).

**Initialization**: For all destination NTPCs, we set the $L$ labels to zero, and the $P$ labels to the disutility corresponding to the arrival time, as shown in Figure B.2. Also, the appropriate $\rho$ label is set to unity, and the others to zero. At all other NTPCs, the $L$ labels are set to infinity, and the $\rho$ labels to zero. SEL is initialized to contain all node-time pairs that can reach the destination by traversing only one arc.

**Iteration 1**: Remove node-time pair (A,2) from SEL. Since the only way to reach (A,2) is via arc 2 in state 1, in the outer *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* Calculate the $\rho$ and $L$ labels using the formulas (4.2) and (4.3); however, as the downstream $\rho$ and $L$ labels are all $\infty$, no improvement is found, so no labels are updated and no node-time pairs are added to SEL.

*Arc 5*: We first calculate the $\rho$ labels:

$\rho_5(A,2,2,1,1) = (0.8)(1) + (0.2)(1) = 1$

$\rho_5(A,2,2,1,2) = (0.8)(0) + (0.2)(0) = 0$

Then we use formulas (4.3) and (4.2) to find the $L$ labels:

$$L_5(A,2,2,1,1,1) = (1)(0.8)(1)(1)^1(1)/(1) + (1)(0.8)(1)(1)^0(0)/(1) + \\ + (1)(0.2)(1)(2)^1(1)/(1) + (1)(0.2)(1)(2)^0(0)/(1)$$

$$= 1.2$$

$L_5$(A,2,2,1,2,1) and $L_5$(A,2,2,1,2,2) are both found to be zero since $\rho_5$(A,2,2,1,2) is zero. That is, since the probability of arriving in the interval (4, ∞) is zero, it is meaningless to find moments of travel time conditional on arrival during this interval. (Recall that we defined this conditional probability to be zero in Section 4.4.)

We now find the expected disutility of following arc 5 using Corollary 4.2 and the identity $E\left[\left(X+t\right)^n \middle| C_s, A_j\right] = \sum_{k=0}^{n} \binom{n}{k} t^{n-k} E\left[X^k \middle| C_s, A_j\right]$ from Section 4.5.1:

$$P_5(\text{A},2,2,1) = (1)(4)(1)(2)^0(1)$$
$$+ (1)(-1)(1)(2)^1(1) + (1)(-1)(1)(2)^0(1.2)$$
$$= 0.8$$

excluding the terms corresponding to $j = 2$ because $\rho_5$(A,2,2,1,2) = 0.

Since $0.8 < \infty$ we set the labels at (A,2,2,1) to these temporary values, and set the path pointer $\pi$ to arc 5. Node-time pair (O,0) is the only predecessor of (A,2), and is added to SEL.

**Iteration 2**: Remove node-time pair (B,3) from SEL. Since the only way to reach (B,3) is via arc 3 in state 1, and the only outgoing arc is arc 4, the outer *for* loop consists only of a single evaluation:

*Arc 4:* We first calculate the $\rho$ labels, then use these to find the $L$ labels:

$\rho_4(B,3,3,1,1) = (1)(1) = 1$

$\rho_4(B,3,3,1,2) = (1)(0) = 0$

$L_4(B,3,3,1,1,1) = (1)(1)(1)(1)^1(1)/(1) + (1)(1)(1)(1)^0(0)/(1) = 1$

$L_4(B,3,3,1,2,1)$ and $L_4(B,3,3,1,2,2)$ are both found to be zero since $\rho_4(B,3,3,1,2)$ is zero.

Now we calculate $P$:

$P_4(B,3,3,1) = (1)(4)(1)(3)^0(1)$

$+ (1)(-1)(1)(3)^1(1) + (1)(-1)(1)(3)^0(1)$

$= 0$

excluding the terms corresponding to $j = 2$ because $\rho_5(B,3,3,1,2) = 0$.

Since $0 < \infty$ we set the labels at (B,3,3,1) to these temporary values, and set the path pointer $\pi$ to arc 4. Node-time pair (O,0) is a predecessor of (B,3), but is already in SEL; node-time pair (A,2) is a predecessor of (B,3) that has already been scanned, but is no longer in SEL. Therefore, we must add (A,2) to SEL.

**Iteration 3**: Remove node-time pair (A,2) from SEL. (This is the second time it has been scanned.) As before, in the outer *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* Proceeding in the same manner, we find

$\rho_3(A,2,2,1,1) = (1)(1) = 1$

$\rho_3(A,2,2,1,2) = (1)(0) = 0$

$L_3(A,2,2,1,1,1) = (1)(1)(1)(1)^1(1)/(1) + (1)(1)(1)(1)^0(1)/(1) = 2$

$L_3(A,2,2,1,2,1)$ and $L_4(A,2,2,1,2,2)$ are both found to be zero since $\rho_4(B,3,3,1,2)$ is zero.

$P_4(B,2,2,1) = (1)(4)(1)(2)^0(1)$

$+ (1)(-1)(1)(2)^1(1) + (1)(-1)(1)(2)^0(2)$

$= 0$

Since $0 < 0.8$ we set the labels at (A,2,2,1) to these temporary values, and update the path pointer $\pi$ to arc 3. That is, after step 1, arc 5 was the preferred arc when arriving at node A at time 2, but in this step, choosing arc 3 was seen to produce a lower disutility, so we update the policy accordingly. Node-time pair (O,0) is the only predecessor of (A,2), and is already in SEL.

*Arc 5*: As before,

$\rho_5(A,2,2,1,1) = (0.8)(1) + (0.2)(1) = 1$

$\rho_5(A,2,2,1,2) = (0.8)(0) + (0.2)(0) = 0$

$L_5(A,2,2,1,1,1) = (1)(0.8)(1)(1)^1(1)/(1) + (1)(0.8)(1)(1)^0(0)/(1) +$

$+ (1)(0.2)(1)(2)^1(1)/(1) + (1)(0.2)(1)(2)^0(0)/(1)$

$= 1.2$

$L_5(A,2,2,1,2,1) = 0$

$L_5(A,2,2,1,2,2) = 0$

$P_5(A,2,2,1) = (1)(4)(1)(2)^0(1) + (1)(-1)(1)(2)^1(1) + (1)(-1)(1)(2)^0(1.2) = 0.8$

Since $0.8 > 0$ the labels are not updated, and no node-time pairs are considered for addition to SEL.

**Iteration 4**: Remove node-time pair (A,3) from SEL. As the only possible combination of predecessor arc and predecessor state is arc 2 in state 2, in the outer *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:* The downstream labels are all infinity, so following arc 3 will not improve the routing policy. No labels are updated and SEL is not changed.

*Arc 5*:

$\rho_5(A,3,2,2,1) = (0.2)(1) + (0.8)(0) = 0.2$

$\rho_5(A,3,2,2,2) = (0.2)(0) + (0.8)(1) = 0.8$

$L_5(A,3,2,2,1,1) = (1)(0.2)(1)(1)^1(1)/(0.2) + (1)(0.2)(1)(1)^0(0)/(0.2) +$

$+ (1)(0.8)(0)(2)^1(1)/(0.2) + (1)(0.8)(0)(2)^0(0)/(0.2)$

$= 1$

$L_5(A,3,2,2,2,1) = (1)(0.2)(0)(1)^1(1)/(0.8) + (1)(0.2)(0)(1)^0(1)/(0.8) +$

$+ (1)(0.8)(1)(2)^1(1)/(0.8) + (1)(0.8)(0)(2)^0(0)/(0.8)$

$= 2$

$$L_5(A,3,2,2,2,2) = (1)(0.2)(0)(1)^2(1)/(0.8) + (2)(0.2)(0)(1)^1(0)/(0.8) +$$

$$+ (1)(0.2)(0)(1)^0(0)/(0.8) +$$

$$+ (1)(0.8)(1)(2)^2(1)/(0.8) + (2)(0.8)(1)(2)^1(0)/(0.8) +$$

$$+ (1)(0.2)(0)(2)^0(0)/(0.8)$$

$$= 4$$

$$P_5(A,3,2,2) = (1)(4)(0.2)(3)^0(1) +$$

$$+ (1)(-1)(0.2)(3)^1(1) + (1)(-1)(0.2)(3)^0(1) +$$

$$+ (1)(48)(0.8)(3)^0(1)$$

$$+ (1)(-24)(0.8)(3)^1(1) + (1)(-24)(0.8)(3)^0(2) +$$

$$+ (1)(3)(0.8)(3)^2(1) + (1)(3)(0.8)(3)^1(2) + (1)(3)(0.8)(3)^0(4)$$

$$= 2.4$$

Since $2.4 < \infty$ the labels are updated, and the path pointer is set to arc 5. The only predecessor is (O,0) which is already in SEL.

**Iteration 5**: Remove node-time pair (B,5) from SEL. As in Iteration 2, the outer *for* loop consists of a single evaluation:

$\rho_4(B,5,3,2,1) = (1)(0) = 0$

$\rho_4(B,5,3,2,2) = (1)(1) = 1$

$L_4(B,5,3,2,1,1) = 0$ since $\rho_4(B,3,3,1,1)$ is zero.

$L_4(B,5,3,2,2,1) = (1)(1)(1)(1)^1(1)/(1) + (1)(1)(1)(1)^0(0)/(1) = 1$

$L_4(B,5,3,2,2,2) = (1)(1)(1)(1)^2(1)/(1) + (1)(1)(1)(1)^1(0)/(1) + (1)(1)(1)(1)^0(0)/(1) =$

$= 1$

$P_4(B,5,3,2) = (1)(48)(1)(5)^0(1)$

$+ (1)(-24)(1)(5)^1(1) + (1)(-24)(1)(5)^0(1)$

$+ (1)(3)(1)(5)^2(1) + (1)(3)(1)(5)^1(1) + (1)(3)(1)(5)^0(1)$

$= 12$

excluding the terms corresponding to $j = 1$ because $\rho_5(B,5,3,2,1) = 0$.

Since $12 < \infty$ the labels are updated, and the path pointer is set to arc 4. Predecessor (O,0) is already in SEL; predecessor (A,3) is added to SEL for the second time.

**Iteration 6**: Remove node-time pair (A,3) from SEL. As in Iteration 4, in the outer *for* loop we only need to iterate over the two outgoing arcs, 3 and 5.

*Arc 3:*

$\rho_3(A,3,2,2,1) = (1)(0) = 0$

$\rho_3(A,3,2,2,2) = (1)(1) = 1$

$L_3(A,3,2,2,1,1) = 0$ since $\rho_3(A,3,2,2,1)$ is zero.

$L_3(A,3,2,2,2,1) = (1)(1)(1)(2)^1(1)/(1) + (1)(1)(1)(2)^0(0)/(1) = 2$

$L_3(A,3,2,2,2,2) = (1)(1)(1)(2)^2(1)/(1) + (1)(1)(1)(2)^1(0)/(1) + (1)(1)(1)(2)^0(0)/(1) =$

$= 4$

$P_3(\text{A},3,2,2) = (1)(48)(1)(3)^0(1)$

$+ (1)(-24)(1)(3)^1(1) + (1)(-24)(1)(3)^0(2)$

$+ (1)(3)(1)(3)^2(1) + (1)(3)(1)(3)^1(2) + (1)(3)(1)(3)^0(4)$

$= 12$

excluding the terms corresponding to $j = 1$ because $\rho_5(\text{B},5,3,2,1) = 0$.

The current disutility label at (A,3,2,2) is 2.4. Since 12 > 2.4, no updating occurs. That is, arc 5 remains the superior choice at this NTPC.

*Arc 5*: These temporary labels are found to be the same as before, so again, no updating occurs.

**Iteration 7**: Remove (B,2) from SEL. As in Iteration 2, the outer *for* loop consists of a single evaluation:

$\rho_4(\text{B},2,1,1,1) = (1)(1) = 1$

$\rho_4(\text{B},2,1,1,2) = (1)(0) = 0$

$L_4(\text{B},2,1,1,1,1) = (1)(1)(1)(1)^1(1)/(1) + (1)(1)(1)(1)^0(0)/(1) = 1$

$L_4(\text{B},2,1,1,2,1) = 0$ since $\rho_4(\text{B},2,1,1,2)$ is zero.

$L_4(\text{B},2,1,1,2,2) = 0$ since $\rho_4(\text{B},2,1,1,2)$ is zero.

$P_4(\text{B},2,1,1) = (1)(4)(1)(2)^0(1) + (1)(-1)(1)(2)^1(1) + (1)(-1)(1)(2)^0(1) = 1$

Since $1 < \infty$ the labels are updated, and the path pointer is set to arc 4. The only predecessor, (O,0), is already in SEL.

**Iteration 8**: Remove (O,0) from SEL. We calculate the labels as usual for the two choices of outgoing arc:

*Arc 1:*

$\rho_1(O,0,\varnothing,0,1) = (1)(1) = 1$

$\rho_1(O,0,\varnothing,0,2) = (1)(0) = 0$

$L_1(O,0,\varnothing,0,1,1) = (1)(1)(1)(2)^1(1)/(1) + (1)(1)(1)(2)^0(1)/(1) = 3$

$L_1(O,0,\varnothing,0,2,1) = 0$ since $\rho_1(O,0,\varnothing,0,2)$ is zero.

$L_1(O,0,\varnothing,0,2,2) = 0$ since $\rho_1(O,0,\varnothing,0,2)$ is zero.

$P_1(O,0,\varnothing,0) = (1)(4)(1)(0)^0(1) + (1)(-1)(1)(0)^1(1) + (1)(-1)(1)(0)^0(3) = 1$

Since $1 < \infty$ we update labels at the origin and set the path pointer to arc 1.

*Arc 2:*

$\rho_2(O,0,\varnothing,0,1) = (0.5)(1) + (0.5)(0.2) = 0.6$

$\rho_2(O,0,\varnothing,0,2) = (0.5)(0) + (0.5)(0.8) = 0.4$

$L_2(O,0,\varnothing,0,1,1) = (1)(0.5)(1)(2)^1(1)/(0.6) + (1)(0.5)(1)(2)^0(2)/(0.6) +$

$+ (1)(0.5)(0.2)(3)^1(1)/(0.6) + (1)(0.5)(0.2)(3)^0(1)/(0.6) = 4$

$L_2(O,0,\varnothing,0,2,1) = (1)(0.5)(0)(2)^1(1)/(0.4) + (1)(0.5)(0)(2)^0(2)/(0.4) +$

$+ (1)(0.5)(0.8)(3)^1(1)/(0.4) + (1)(0.5)(0.8)(3)^0(1)/(0.4) = 5$

$$L_2(O,0,\varnothing,0,2,2) = (1)(0.5)(0)(2)^2(1)/(0.4) + (2)(0.5)(0)(2)^1(2)/(0.4) +$$

$$+ (1)(0.5)(0)(2)^0(4)/(0.4) +$$

$$+ (1)(0.5)(0.8)(3)^2(1)/(0.4) + (2)(0.5)(0.8)(3)^1(2)/(0.4) +$$

$$+ (1)(0.5)(0.8)(3)^0(4)/(0.4)$$

$$= 25$$

$$P_2(O,0,\varnothing,0) = (1)(4)(0.6)(0)^0(1) +$$

$$+ (1)(-1)(0.6)(0)^1(1) + (1)(-1)(0.6)(0)^0(4) +$$

$$+ (1)(48)(0.4)(0)^0(1) +$$

$$+ (1)(-24)(0.4)(0)^1(1) + (1)(-24)(0.4)(0)^0(5) +$$

$$+ (1)(3)(0.4)(0)^2(1) + (2)(3)(0.4)(0)^1(5) + (1)(3)(0.4)(0)^0(25)$$

$$= 1.2$$

Since $1.2 > 1$, choosing arc 1 is superior to choosing arc 2, so we do not update any labels. At this point SEL is empty, and we terminate the algorithm.

Table B.2 shows the labels and contents of SEL at each step of the algorithm, as discussed in detail above. Table B.3 displays the final routing policy, by showing the choice of outgoing node at each NTPC (excluding the destination NTPCs). Table B.4 lists each realization of the network, along with the probability that it occurs when following the policy in Table B.3.

**Table B.2.** Labels and SEL at each iteration of the algorithm.

| Step: | Initialize | | | | | | | 1 | | | | | | | 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scan: | N/A | | | | | | | (A,2) | | | | | | | (B,3) | | | | | | |
| NTPC | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ |
| (O,0,∅,0) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (A,2,2,1) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 1.2 | 1 | 0 | 0 | 0 | 0.8 | 5 | 1.2 | 1 | 0 | 0 | 0 | 0.8 | 5 |
| (A,3,2,2) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (B,2,1,1) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (B,3,3,1) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 1 | 1 | 0 | 0 | 0 | 0 | 4 |
| (B,5,3,2) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (D,3,4,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,3,5,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,4,4,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,2) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,5,5,2) | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – |
| (D,6,4,1) | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – |
| SEL: | (B,1)<br>(B,3)<br>(B,5)<br>(A,2)<br>(A,3) | | | | | | | (B,1)<br>(B,3)<br>(B,5)<br>(A,2)<br>(O,0) | | | | | | | (B,1)<br>(B,5)<br>(A,3)<br>(O,0)<br>(A,2) | | | | | | |

| Step: | 3 | | | | | | | 4 | | | | | | | 5 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scan: | (A,2) | | | | | | | (A,3) | | | | | | | (B,5) | | | | | | |
| NTPC | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ |
| (O,0,∅,0) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (A,2,2,1) | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 3 |
| (A,3,2,2) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 1 | 0.2 | 2 | 4 | 0.8 | 2.4 | 5 | 1 | 0.2 | 2 | 4 | 0.8 | 2.4 | 5 |
| (B,2,1,1) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – |
| (B,3,3,1) | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 4 |
| (B,5,3,2) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 0 | 0 | 1 | 1 | 1 | 12 | 4 |
| (D,3,4,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,3,5,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,4,4,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,2) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,5,5,2) | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – |
| (D,6,4,1) | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – |
| SEL: | (B,1)<br>(B,5)<br>(A,3)<br>(O,0) | | | | | | | (B,1)<br>(B,5)<br>(O,0) | | | | | | | (B,1)<br>(O,0)<br>(A,3) | | | | | | |

| Step: | 6 | | | | | | | 7 | | | | | | | 8 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scan: | (A,3) | | | | | | | (B,1) | | | | | | | (O,0) | | | | | | |
| NTPC | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ | $L_{11}$ | $\rho_1$ | $L_{21}$ | $L_{22}$ | $\rho_2$ | P | $\pi$ |
| (O,0,∅,0) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 3 | 1 | 0 | 0 | 0 | 1 | 1 |
| (A,2,2,1) | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 3 |
| (A,3,2,2) | 1 | 0.2 | 2 | 4 | 0.8 | 2.4 | 5 | 1 | 0.2 | 2 | 4 | 0.8 | 2.4 | 5 | 1 | 0.2 | 2 | 4 | 0.8 | 2.4 | 5 |
| (B,2,1,1) | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | – | 1 | 1 | 0 | 0 | 0 | 1 | 4 | 1 | 1 | 0 | 0 | 0 | 1 | 4 |
| (B,3,3,1) | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 4 |
| (B,5,3,2) | 0 | 0 | 1 | 1 | 1 | 12 | 4 | 0 | 0 | 1 | 1 | 1 | 12 | 4 | 0 | 0 | 1 | 1 | 1 | 12 | 4 |
| (D,3,4,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,3,5,1) | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – | 0 | 1 | 0 | 0 | 0 | 1 | – |
| (D,4,4,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,1) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,4,5,2) | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – | 0 | 1 | 0 | 0 | 0 | 0 | – |
| (D,5,5,2) | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – | 0 | 0 | 0 | 0 | 1 | 3 | – |
| (D,6,4,1) | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – | 0 | 0 | 0 | 0 | 1 | 12 | – |
| SEL: | (B,1)<br>(O,0) | | | | | | | (O,0) | | | | | | | empty | | | | | | |

**Table B.3.** Minimum deviance policy for this network.

| NTPC | Choice of arc |
|---|---|
| (O,0,∅,0) | 1 |
| (A,2,2,1) | 3 |
| (A,3,2,2) | 5 |
| (B,2,1,1) | 4 |
| (B,3,3,1) | 4 |
| (B,5,3,2) | 4 |

**Table B.4.** Probability of each realization occuring under this policy.

| Realization | Cost | Probability |
|---|---|---|
| (O,0,∅,0), (B,2,1,1), (D,3,4,1) | 2 | 1 |
| (O,0,∅,0), (A,2,2,1), (B,3,3,1), (D,4,4,1) | 4 | 0 |
| (O,0,∅,0), (A,2,2,1), (D,3,4,1) | 3 | 0 |
| (O,0,∅,0), (A,2,2,1), (D,4,4,2) | 4 | 0 |
| (O,0,∅,0), (A,3,2,2), (B,5,3,1), (D,6,4,1) | 6 | 0 |
| (O,0,∅,0), (A,3,2,2), (D,4,4,1) | 4 | 0 |
| (O,0,∅,0), (A,3,2,2), (D,5,4,2) | 5 | 0 |

# References


Ahuja, R., T. Magnanti, and J. Orlin. (1993) *Network flows*. Prentice-Hall, Englewood, NJ.

Andreatta, G. and L. Romeo. (1988) Stochastic shortest paths with recourse. *Networks* 18, 193-204.

Bellman, R. (1957) *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Chabini, I. (1998) Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record* 1645, 170-175.

Dijkstra, E. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 269-271.

Dreyfus, S. (1969) An appraisal of some shortest-path algorithms. *Operations Research* 17, 395-412.

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

Fan, Y., R. Kalaba, and J. Moore. (2005) Arriving on time. *Journal of Optimization Theory and Applications* 127, 497-513.

Ford, L. (1956) Network flow theory. Report P-923, Rand Corp., Santa Monica, CA.

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

Fu, L. and L. Rilett. (1998) Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research B* 32 (1998), 499-516.

Gabriel, S. and D. Bernstein. (1997) The traffic equilibrium problem with nonadditive path costs. *Transportation Science* 31 (1997), 337-348.

Gabriel, S. and D. Bernstein. (1999) Nonadditive shortest paths. NJTIDE report.

Gao, S. (2005) Optimal adaptive routing and traffic assignment in stochastic time-dependent networks. Ph.D. dissertation, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology.

Gao, S. and I. Chabini. (2006) Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research B* 40, 93-122.

Hall, R. (1986) The fastest path through a network with random time-dependent travel times. *Transportation Science* 20, 182-188.

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

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

Miller-Hooks, E. and H. Mahmassani. (2000) Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science* 34, 198-215.

Mirchandani, P. (1976) Shortest distance and reliability of probablistic networks. *Computers and Operations Research* 3, 347-355.

Montemanni, R. and L. Gambardella. (2004) An exact algorithm for the robust shortest path problem with interval data. *Computers and Operations Research* 31, 1667-1680.

Murthy, I. and S. Sarkar. (1996) A relaxation-based pruning technique for a class of stochastic shortest path problems. *Transportation Science* 30, 220-236.

Pape, U. (1974) Implementation and efficiency of Moore-algorithms for the shortest route problem. *Mathematical Programming* 7, 212-222.

Polychronopoulos, G. and J. Tsitsiklis. (1996) Stochastic shortest path problems with recourse. *Networks* 27, 133-143.

Prékopa, A. (1990) The discrete moment problem and linear programming. *Discrete Applied Mathematics* 27, 235-254.

Provan, J. (2003) A polynomial-time algorithm to find shortest paths with recourse. *Networks* 41, 115-125.

Psaraftis, H. and J. Tsitsiklis. (1993) Dynamic shortest paths in acyclic networks with Markovian arc costs. *Operations Research* 41, 91-101.

Redmond, L. and P. Mokhtarian. (2001) The positive utility of the commute: modeling ideal commute time and relative desired commute amount.". *Transportation* 28, 179-205.

Sigal, C., A. Pritsker, and J. Solberg. (1980) The stochastic shortest route problem. *Operations Research* 28, 1122-1129.

Sivakumar, R. and R. Batta. (1994) The variance-constrained shortest path problem. *Transportation Science* 28, 309-316.

Tsaggouris, G. and Zaroliagis, C. (2004) Non-additive shortest paths. In Albers, S. and Radzik, T. (eds.), *Proc., 12th Annual European Symposium on Algorithms*, 822-834.

Waller, S. and A. Ziliaskopoulos. (2002) On the online shortest path problem with limited arc cost dependencies. *Networks* 40, 216-227.

Yu, G. and J. Yiang. (1998) On the robust shortest path problem. *Computers and Operations Research* 25, 457-468.

Ziliaskopolous, A. and H. Mahmassani. (1993) Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record* 1408, 94-100.

# Vita

Stephen David Boyles was born in Honolulu, Hawaii on October 19, 1982 to Jean Hom Boyles (*née* Jean Lee Hom) and David Franklyn Boyles. After graduating from Rogers High School in Puyallup, Washington in 2000, he enrolled at the University of Washington in Seattle, Washington, where he receieved Bachelor of Science degrees in civil engineering and mathematics, graduating *magna cum laude* in 2004. In August 2004, he entered The Graduate School at The University of Texas at Austin.

Permanent Address: 8813 164th St E

Puyallup, Washington 98375-2005

This thesis was typed by the author.