THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

A MEAN-VARIANCE MODEL
FOR STOCHASTIC TIME-DEPENDENT NETWORKS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

JAEJIN SEOK
Norman, Oklahoma
2005

A MEAN-VARIANCE MODEL
FOR STOCHASTIC TIME-DEPENDENT NETWORKS

A Dissertation APPROVED FOR THE
SCHOOL OF INDUSTRIAL ENGINEERING

BY

[illegible]

Dr. P. Simin Pulat, Committee Chair

[illegible]

Dr. Thomas Landers

[illegible]

Dr. Sridhar Radhakrishnan

Teri Reed Rhoads

Dr. Teri Reed Rhoads

[illegible]

Dr. Theodore B. Trafalis

Acknowledgements

Praise God from whom all blessings flow! First, I would like to thank God for the constant inspiration, spirit, and faith which have provided me with what I most needed to complete this study. With God all things are possible and thus it has been so for me. Amen.

This dissertation would not have been possible without many people who have contributed to my life and research. I owe them all my sincerest gratitude. My advisor, Dr. Pulat has provided me with the thoughtful guidance support, and direction for this dissertation. She has been a guide when I felt confused in a maze of details, always and consistently keeping perspectives on the bigger picture. She will always be appreciated and remembered. I would like to express my appreciation to the other committee members, Dr. Landers, Dr. Radhakrishnan, Dr. Rhoads, and Dr. Trafalis for their valuable insights and comments during the progress of this research.

This work is dedicated to my parents and all family members who supported and encouraged me not only during my study at OU but throughout my life. Specially, I thank my wife, Hyunsook, and my beautiful daughters, Jisun and Hayoung, for their patience and understanding for the many hours I have stolen from their lives. Without their sacrifice, I would never have completed my research.

I give special thanks to my brother in law’s family who truly have helped my family in many way. I also would like to express my appreciation to all of church members for their prayer support.

Table of Contents

List of Figures

List of Tables

Abstract

Traditional models of route generation are based on choosing routes that minimize expected travel-time between origin and destination. The variance of the least-time path is not included in the path selection. In addition, due to congestion in transportation networks, travel times are time-dependent and stochastic in nature. This research focuses on the time dependency as well as the stochastic nature of traffic flow.

Two algorithms are developed for determining a minimum travel time variance path and minimum mean-variance path assuming a priori best path routing policy. Under this policy, drivers use the path that corresponds to the minimum travel time variance to their destination node determined prior to the actual departure time at an origin node. We prove that both algorithms reach the optimal solution in finite number of steps but have non-polynomial running times. In addition, two algorithms, specialized modified label correcting and label setting algorithms, are developed for determining minimum mean-variance travel time path for time-adaptive routing problem. These algorithms allow the travel to define the route as he/she travels from the origin to the destination. Both algorithms reach optimal solution in finite number of steps and have polynomial computational complexity.

The computational performance of the algorithms was evaluated through numerical experiments using randomly generated networks. A regression curve relating the running time to number of nodes, arc density, number of time intervals, and the number of discrete arc travel times has been generated for each algorithm. The results show that number of nodes and arc density influence the running time worse than linearly. The proposed algorithms were illustrated using a real-life network and near-real time travel information between Beverly Hills and Garden Grove in Los Angeles, California. The data was generated using the Freeway Performance Measurement System (PaMS) run by California Department of Transportation and the University of California at Berkeley. The illustration showed that more research is needed in extracting travel time information from real-life data which is vast and influenced by several factors such as the day of the week, holidays, time of the day, accidents. However, through the illustration we were able to demonstrate how the proposed algorithms can be used with near real-time information.

Chapter 1. Introduction

1.1 Motivation

Time is the essence in today's ever-mobile world. While the time it takes to travel long distances is getting shorter each year, daily commuters spend more time just to get to work. One main reason for this situation is traffic congestion. Traffic congestion is perhaps the most prominent problem of modern society and has both direct and indirect impact. As described in the 2005 Urban Mobility Report, by the Texas Institute of Transportation: “Despite a slow growth in jobs and travel in 2003, congestion caused 3.7 billion hours of travel delay and 2.3 billion gallons of wasted fuel, an increase of 79 million hours and 69 million gallons from 2002 to a total cost of more than $63 billion.” To an individual traveler, congestion reduces the quality of life by reducing leisure time, increasing anxiety, and wasting personal resources. To firms, congestion reduces the work efficiency of employees and increases freight transportation costs. To the society as a whole, congestion negatively affects environmental quality by causing more gas emissions and noise, and endangers traffic safety by causing stress and fatigue for drivers.

Adding capacity by building infrastructure is the most common measure taken to improve traffic flow. However, the more traditional approach of simply adding more infrastructure and capacity is not always possible or desirable. Furthermore, new infrastructure will induce more demand, which could affect the increased capacity or even make the congestion worse. However, improvements can still be made by increasing the efficiency of the existing system. These treatments are particularly effective in three ways. They have a relatively low cost. They can usually be implemented quickly and can be tailored to individual situations, making them more useful because they are flexible. They are usually a distinct, visible change; it is obvious that the operating agencies are reacting to the situation and attempting improvements.

Measures to relieve traffic congestion are generally based on the concept of making the best use of current infrastructure with the advanced information technology, which is the underlying idea of Intelligent Transportation Systems (ITS). Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS) are fields of ITS which put congestion management as a major priority. Among the various sub-systems of ITS, ATIS aims to provide travelers with updated and useful information about network conditions to traveler to aid in making better decision. As traffic conditions are stochastic in nature, the information is of even greater importance. For example, when an accident happens, a timely notice by ATIS to travelers who plan to take the route on which the accident happened would be quite beneficial.

All parameters of a real time traffic such as traffic volume and travel time are stochastic in nature. The travel time of even commonly undertaken trips like home to work differ over day of the week and even time of the day. The randomness can come from multiple sources that are both recurrent and non-recurrent. One of the most significant sources is the disturbances that cause unpredictable (called non-recurrent) congestion, such as accidents and vehicle breakdown. Traffic conditions with predictable (recurrent) congestion, on the other hand, are also usually different from day to day, largely because of fluctuations in origin-destination (OD) trips. The fluctuations can be in both the total number of OD trips and the spread of OD trips over departure times (i.e.

traffic flow per unit time). Travelers with non-commuting trip purposes might decide not to take a trip on a particular day, due to other personal business, and the no-travel decisions collectively result in a random number of OD trips. Travelers may also respond to congestion by shifting departure times from day to day, and thus, there exists a random pattern in OD trips' spread. These and more reasons make the traffic flow more stochastic and less predictable.

Travelers make decisions (destination, mode, departure time, route) based on their information about the traffic network. The information can be obtained through a wide range of means: the travelers' own experience, word of mouth, radio broadcast, variable message signs (VMS), an in-vehicle communication system, and so on. This information can be classified as a priori or real time. A priori information is about the day-to-day fluctuations of traffic quantities, e.g. the time it takes to travel from Norman to OKC is 20 minutes on average, but roughly once in a month, the travel time is unusually high, due to various reasons. Real time information is about the traffic conditions on a specific day, e.g. an incident just occurred on a route, and thus will impact the traffic for the next 30 minutes. This classification is meaningful only when there is stochasticity in the network; it is in that way real time information is different from a priori information. Destination, departure time and mode decisions are usually made only at origins and can rarely be changed en route, while route decisions can be changed en route more easily and thus benefit more from real time information. ATIS can provide both a priori and real time information. Travelers only have personal experience to guide them on their selected routes. In order to obtain a priori information about the whole network, they need to go beyond their personal experience, and a good source is ATIS. ATIS can provide travelers with reports of traffic conditions in the past and possibly predictions about the future, for the temporal and spatial ranges and in formats specified by travelers. Combining all sources of priori information, travelers can form their own general pictures about the network. Nevertheless, the benefit of ATIS is primarily embodied through the provision of real time information, especially in a network disturbed randomly by accidents, vehicle breakdowns, bad weather, work zones, special events, and so on. The sensors are, in general, installed on state highways and interstates to collect real time traffic data. This real time sensor data can be use to assist commuters (passengers, emergency and commercial vehicles) in making the best decisions on route selection.

Travelers' routing decisions in a stochastic network with real time information is conceivably different from those in a deterministic network. It is generally believed that adaptive routing will save travel time and enhance travel time reliability. For example, in a network with random incidents, if one does not adapt to an incident scenario, he/she could be delayed in the incident link for a very long time. However, if adequate real time information is available about the incident and the traveler adapts to it by taking an alternative route, he/she can save travel time as compared to the non-adaptive case. The adaptiveness also ensures that the travel time is not prohibitively high in incident scenarios, and thus provides a more reliable travel time.

It is therefore a very interesting research question how an individual traveler makes adaptive routing decisions in a stochastic and time-dependent network. Traditional models of route generation are based on choosing routes that minimize expected travel-time between origin and destination. Such approaches do not account for the fact that travelers often incorporate travel-time variability within their decision making. Thus, a route with lower travel-time variability is preferred by some travelers, even if such a route is not one with the lowest mean of travel-time. Such traveler behavior is best captured by a multi-objective model (mean-variance) in which the choice of a route is based on the mean as well as the variance of the path's travel-time. The route planning model developed in this work is intended to help travelers with decision making.

1.2 Research Overview

1.2.1 Problem Statement: Stochastic Time-Dependent Networks

Travel time between an origin and destination is often the primary criterion in routing vehicles such as ambulances, police cars, vehicles carrying hazardous substances, and individuals on their way to some activity. Today's traffic routes have ever-changing traffic patterns that lead to time-varying dynamic networks. To analyze this dynamic situation, one would need to use dynamic algorithms that can take into account time-dependent behavior. This leads to the concept of dynamic network optimization and real-time traffic routing.

There may be some uncertainty associated with the travel times (or travel costs) along the arcs as a result of inherent uncertainties in future trip times, incomplete a priori information, or inaccurate methods of predicting future trip times. In part, such inherent uncertainties in travel times can be attributed to the varying characteristics of drivers and vehicles, the amount of interaction between vehicles due to the level of congestion and the unexpected delays as a result of automobile failures or accidents, construction or lane closures, road hazards, train passings, bus stops, and so on.   In order to optimally route vehicles, both the stochastic and time-dependent nature of the travel times must be considered. Future travel times, as well as other travel costs, can be treated as random variables whose probability distribution functions vary with time.

Two types of algorithms are generally used for routing in networks: (1) shortest path based routing algorithms, and (2) optimal routing algorithms based on other measures. The efficiency of a routing algorithm depends on how it performs during times of congestion in the network. The main tasks that have to be performed by these routing algorithms are routing choice and the error-free and reliable delivery of a message.

The average travel-time that would be experienced by drivers may depend on the type of routing policy adopted. In literature, the following two routing policies have been considered (Chabini, 2001).

1) The best path routing policy (Route planning): A minimum of the expected travel-time path is determined in this policy during the drivers' trips from an origin node to a destination node. Because of travel time random variable, there can be exist multiple criteria to measure the quality of a path. The expected travel time is only one possible criterion.
2) The best next-arc routing policy (Route guidance): Rather than determining a single best path based only on information known before travel begins, routes with lower travel times may be obtained by allowing the driver to react en route to revealed (actual) arrival times. This is referred to as time-adaptive route choice by Hall (1986). A driver recursively selects the best next arc to follow when departing from the current node, depending on the actual arrival time at this node.

These two policies possess the same solution in static or in deterministic problems. However, in general stochastic time-dependent networks, the above two policies lead to different expected travel-times. There are two results that follow from this interpretation: (1) the expected travel time corresponding to the second policy is always less than or equal to the expected travel time corresponding to the first policy, and (2) generating a solution to the second policy is a less difficult algorithmic task than computing a solution to the first one (Chabini, 2001). The aim of the best next arc routing policy is to compute an a priori routing strategy that would be repetitively used to guide drivers during their trips between a pair of origin-destination nodes. Under the best path routing policy, drivers use the same path. Under the best next-arc routing policy, drivers may follow different paths.

1.2.3 Solution difficulties and General Approaches

Many difficulties come up to solve the problem of determining "best" paths in stochastic, time-dependent networks that are not present in the deterministic, time-dependent problem. In problems involving a single objective function, like minimum travel times, of deterministic quantities, a single optimal solution can be identified, with possible alternate optimal solutions. If alternate optimal solutions exist, the solutions all have the same deterministic value; and thus, a solution can be arbitrarily selected without regret. A single objective problem of a stochastic quantity may result in multiple non-dominated solutions because several solutions may have some probability of being best for one or more realizations of the random quantity. For this reason, multiple non-dominated least time (or shortest) paths may exist in stochastic networks where more than one path may have the least time for some realization of the network.

There are numerous criteria that one uses to select one route over another in stochastic time dependent networks. For example, one may prefer the path that has some probability of having the least possible time. Miller-Hooks and Mahmassani (1998a) proposed efficient procedures for determining the least possible time paths in stochastic time dependent networks. In stochastic time dependent networks, paths comparison is further complicated by the fact that such comparisons of the path probability distribution functions must be made over a time period. Several algorithms for determining a priori paths in stochastic time dependent networks that employ such path-comparison techniques were proposed by Miller-Hooks and Mahmassani (1998a, 1998b, and 2000). Common approaches to stochastic problems that often lead to solutions with the least-expected value. The determination of the least-expected time paths in stochastic time dependent networks is more difficult than in networks where the arc traversal time distributions are time-independent. In stochastic time-dependent networks, one cannot simply set the random arc traversal times to their expected values and solve for the least expected time paths through the use of a deterministic shortest path algorithm, as is possible in time-invariant networks. This is because the expected traversal time on an arc in stochastic time-varing networks depends on the time of arrival at its origin node. Miller-Hooks and Mahmassani (1998a, 2000) and Miller-Hooks (2001) discuss approaches for the time-varying and stochastic transportation and data networks. They propose label-correcting algorithm and a modified label-correcting algorithm to determine a priori least expected time path and a lower bound on least expected time path, respectively. They also illustrate how adaptive least expected time hyperpaths can be determined using an extension of the modified label-correcting algorithm to generate best routing policies conditioned on the node arrival times. Chabini (2001) proposed a dynamic programming approach to determine the least expected travel costs from all nodes to a given destination. Gao and Chabini (2001) studied the best routing policy problem. They provide a comprehensive taxonomy of the problem, based on information access and network statistical dependency. An exact algorithm and four approximations are proposed.

Previous approaches in stochastic time dependent problems do not account for the fact that travelers often incorporate travel-time variability in decision making. Thus, a route with lower travel-time variability is preferred at certain situations like hazardous material shipment, even if such a route is not one with the lowest mean of travel-time. Such traveler behavior is best captured by a mean-variance model in which the choice of a route is based on the mean as well as the variance of the path's travel-time. To the best of author's knowledge, there is no papers in the literature that deals with minimum variance and mean-variance path problems in stochastic time-dependent networks. In this study, we develop the methodology for minimum variance and mean-variance path that accommodates variance and both means and variances within a route guidance model.

Two procedures presented in this study are specialized modified label correcting and efficient specialized label setting algorithms for generating "preferred" paths. The first procedure determines a prior minimum variance and mean-variance paths from all origins to a single destination for each departure time in the peak period. The second procedure determines the "best" next arc routing from all origins to a single destination for each departure time in the peak period. In generic label correcting algorithms for time-independent, deterministic shortest path problems, a single label associated with each node maintains the current shortest time from the node to the destination. The labels are updated until optimality conditions are satisfied. Upon termination, as long as a path exists, a single shortest path (which may be tied for shortest) from all origins to the destination node and the corresponding distances (or times) are known. In deterministic, time-dependent networks, vector labels are associated with each node, maintaining the current shortest distance (or time) from the node to the destination for each time interval in the peak period (Ziliaskopoulos and Mahmassani, 1993). Unlike label correcting algorithms, where the components of all vector labels are temporarily set until termination, after each iteration of the label setting algorithm, where the labels are updated for a specific deprture time, *t*, the component of each vector label associated with *t* is permanently set for all *i*.

Similarly, in stochastic, time-dependent networks, vector labels are maintained from each node to the destination node, the number and contents of which now depend on the specific problem that one is solving. If the problem is to determine the least expected time paths, then the optimality conditions based on expected times are used. If the problem is to determine the least variance paths, then the optimality conditions based on the variance of travel times are used. Again multiple vector labels are associated with each node (as more than one path may have the least expected time for one or more departure time intervals). Each vector label maintains the expected time or variance for its associated path from the node to the destination for all *t*. For each of these problem formulations in stochastic, time-varying networks, until termination, the labels of any Pareto-optimal, or optimal, paths must be maintained over the entire time period.

In this work, problem formulations that permit the use of a single vector label from each node, maintaining single deterministic quantities for each departure time, can result in polynomial time algorithms with a worst-case performance similar to that of the deterministic, time-varying shortest path problem.

1.3 Research summary

The procedures developed in this dissertation for determining "best" paths in stochastic, time-dependent networks are organized in two sets. The first set addresses the problems of generating the “best” paths for both a priori best path routing and time-adaptive best routing problems. An additional procedure for determining a priori minimum variance path and minimum mean-variance paths, and time adaptive minimum mean-variance routing is presented.

1.3.1 Path comparison: three dominance criteria

Three dominance criteria, deterministic dominance, stochastic dominance, and expected value dominance, (Miller-Hooks and Mahmassani, 2003) are considered in the determination of non-dominated (or efficient) paths, described in detail in Chapter 3. In this section, a brief description of the rationale behind each dominance criterion is given.

The first dominance criterion is deterministic dominance. By this criterion, if for a given departure time the highest travel time on the best path is lower than the lowest travel time on the second best path, then the second best path has zero-probability of having a lower travel time than the first. The first path is said to dominate the second for the given departure time. For this departure time, one can choose the first path with certainty that the second path will not be better.

The second dominance criterion, stochastic dominance, is less conservative than the deterministic dominance, possibly resulting in fewer non-dominated (or efficient) paths. Here, for a given departure time, the first path dominates second, and if for all possible travel time values, the probability that the first path's travel time is less than or equal to that value is always greater than the probability that the second path's travel time is less than or equal to this same value.

The third criterion considered uses the expected value to establish dominance. If, for all departure time intervals in the peak period, the first path has lower expected time than the second, the first path dominates the second.

In this research, procedures for generating a priori minimum variance time paths, a priori minimum mean-variance paths, and time adaptive minimum mean-variance paths are developed using the third criteria.

1.3.2 A priori Minimum variance and Mean-Variance routing algorithms

A distinctive feature of a traffic network is the link-wise and time-wise stochastic dependency of link travel times. However, a comprehensive literature review on optimal routing policy problems for minimum variance in stochastic time-dependent networks reveals that no research has considered this important feature of a traffic network. When faced with travel time uncertainty, travelers are also concerned about the reliability of their travel times. Travel time variance is used to represent travel time reliability (Sen, et al., 2001). A routing policy with less travel time variance is viewed as more reliable. For commuters, the desired arrival time in the morning might be some time around the work starting time. For a traveler trying to catch a plane, the desired arrival time might be roughly one hour before the plane's departure. It is generally believed and verified that a constant travel time path is preferred over a high variance path, even if it has a shorter travel time. Since expected travel time is the primary criterion in routing optimization, and reliability measures (variance) are generally secondary, it is necessary to design algorithms that minimize linear combination of expected travel time and variance. The detail descriptions are presented in chapter 5.

Two algorithms, PMV and PMMV, are developed for determining a minimum travel time variance path and minimum mean-variance path for a priori best path routing problem. In this routing policy, drivers use the same path that corresponds the minimum travel time variance to their destination node depending on their actual departure time at an origin node.

We find the recursive relationship between means and variances of a given routing policy starting from two adjacent nodes. The node labels are updated by using the recursive formulation. At termination of either algorithm, the final node labels are the minimum travel time variance or mean-variance from each node to the destination node for departure time *t*. Both algorithms are similar because both mean and variance calculations are required in both procedures. One main difference between these two algorithms is the way to update the node labels. Detail descriptions of the algorithms are in Chapter 4.

1.3.3 Time-adaptive Minimum mean-variance routing algorithms

Rather than selecting a priori single best path based only on information known before travel begins, routes with lower expected travel times and variance may be obtained by allowing the driver to react en route to revealed (actual) arrival times.

Two algorithms, TAMMV1 and TAMMV2, are presented for determining minimum mean-variance travel time path for “best” next arc routing problem. The same recursive formulations are used for these algorithms.

The next node is computed like this: a path with a minimum linear combination of expected travel time and variance from the current node to the destination is computed, and then the first link along this path is followed. When the user arrives at the next node, a new minimum linear combination of expected travel time and variance path is computed and the first link followed. Note that the new path is not necessarily a subpath of the previous one. This routing method is adaptive as a new path is computed each time a new decision node is reached, but it is myopic in the sense that it assumes no future changes in network conditions when computing the next node to take.

1.4 Contributions

The contributions of the thesis in stochastic time-dependent networks are summarized as follows:

1. The specific computational steps are developed to find a priori minimum variance path in stochastic time-dependent networks.
2. The specific computational steps are developed to find a priori minimum mean-variance path in stochastic time-dependent networks.

3. Two different computational steps are developed to find minimum mean-variance paths in stochastic time-dependent networks for a time-adaptive routing problem.
4. Computer programs for these models are prepared, and extensive numerical experiments are conducted to assess the average run time.
5. The proposed algorithms were illustrated using a real-life network and near-real time travel information.

1.5 Organization

The dissertation is arranged in seven chapters. This first chapter describes the problems that are addressed and gives an overview of the general approach that is taken for solving these problems. This is followed by a brief description of the procedures that are developed for solving the problems, the contributions of this work, and finally, the organization of this thesis.

In Chapter 2, we survey the literature on this topic, including deterministic shortest path routing problems, k-shortest path routing problems, multi-objective shortest path routing problems, routing in stochastic static networks, and routing in stochastic time-dependent networks. This survey reveals that there are a number of variants of the optimal routing problem in an stochastic time dependent network.

In Chapter 3, some basic concepts, as they apply to stochastic, time-dependent networks, are defined. Techniques for selecting a best path through compromise from the set of non-dominated solutions are discussed. Three dominance criteria are presented for comparing paths with random travel times whose probability distribution functions vary over time. The conditions are given for comparing two paths at a single departure time and multiple paths over the time period for both a priori path selection and time-adaptive route choice.

In Chapter 4, we develop the PMV and the PMMV algorithms to find a priori minimum variance path and minimum mean-variance path in stochastic time dependent networks. We find the recursive relationship between the mean and the variances of a given routing policy starting from two adjacent nodes. The worst-case computational complexity of the algorithms are discussed

In Chapter 5, the TAMMV1 and the TAMMV2 algorithms are developed to find a minimum variance path in stochastic time dependent networks for time-adaptive routing problem. The worst-case computational complexity of both algorithms is discussed.

In Chapter 6, the procedures of chapters 4 and 5 are implemented and tested on numerous randomly generated networks and a more realistic transportation network. The tests are intended to estimate the average run time. The procedures are tested on randomly generated networks with an average arc density of 2, 4, and 6 with number of nodes 50, 100, and 500. The methodologies used for randomly generating the networks and time-varying probability distribution functions are described. The results of these experiments are presented.

The procedures presented in Chapters 4 and 5 are illustrated on an example problem: the best path selection from Beverly Hills to Garden Grove during rush hours in a representation of the Los Angeles traffic network. The travel times of this network are collected on Freeway Performance Measurement System (PeMS) based on actual distances and varying travel speeds by traffic sensors.

Finally, a summary of the dissertation work and future research directions are discussed in Chapter 7.

Chapter 2. Literature Review

The routing problems in networks have been an important and well researched topic for a long time. We first give a brief introduction to the shortest path problem in deterministic networks, including the well developed static shortest path (SSP) problem and the dynamic shortest path problem. This will be useful to the study of routing problems in stochastic networks. We then proceed to stochastic networks. There are various ways of defining a stochastic network. Most of the problem variants studied in literature assumes that the underlying network is static (not dependent on time). Some other variants studied in the literary work with special cases of dynamic stochastic networks do not represent time explicitly. A limited number of papers have studied the optimal routing in a stochastic time dependent network with specific assumptions. A comprehensive study of the problem is not available in the literature.

2.1 Basic Concept and Classification of Shortest Path Problems

The shortest path problem is one of the most fundamental network optimization problems. It is an important problem by itself for its many applications in the real world. It also important as a sub-problem in other network flow problems. The minimum cost flow and the maximum flow problems all can be solved by finding the shortest paths and augmenting flows along such paths. Lawler (1976), Tarjan (1983), and Ahuja et al. (1993) provided an excellent review of how the shortest path problems can be used in other network problems.

Algorithms for solving the shortest path problem have been studied for a long time. However, advances in the theory of shortest path algorithms are still being made.

Let G(N, A) be a network, where N is the set of nodes and A is the set of links. Each link *(i,j)* has a cost *c(i, j)* and we term a path with minimum cost as the shortest path. The SSP is to find the shortest path from a source node s to a destination node *d*. Dijkstra's algorithm is the most commonly used algorithm to solve the shortest path problem for networks with nonnegative arc costs. Various implementations of Dijkstra's algorithm exist. The most straightforward one is based on the array of data structure and has a running time of $O(n^2)$, where n is the number of nodes. The implementation using a Fibonacci heap can achieve a running time of $O(m + n \log n)$, where m is the number of arcs. This implementation is also currently the best strongly polynomial-time algorithm for solving the shortest path problem. If the network has negative arc costs, more sophisticated algorithms (such as the label-correcting algorithms) are needed. These algorithms basically check whether the optimality conditions $d(i)+c(i,j)\geq d(j), \forall(i,j)\in A$ are satisfied, where the label *d(i)* is the cost for the origin to node *i*. They make necessary changes by changing cost labels until no arc violates this condition. A first-in-first-out (FIFO) queue implementation of the label correcting algorithm has a running time of $O(mn)$.

The existing shortest path problems (SPP) and their extensions may be classified by various criteria. The first classification may be by the number of routes identified. There are two categories: One is a generic shortest path algorithm which identifies a single path, and the other is the so-called k shortest path algorithm which identifies k shortest paths. These shortest path algorithms and extensions usually consider a single attribute such as time, cost, distance or a combination of attributes that are combined into a single generalized cost.

The second classification may be based on whether a single attribute or multiple attributes are considered in the objective function. The majority of the traditional shortest path and k-shortest path algorithms belong to the single attribute category, while the multi-criteria shortest path problem (MCSPP) belongs to the second category. Table 2.1 summarizes the classifications of the SPP and gives examples of each type.

Table 2.1. Classifications of shortest path problems

Classification	**Number of routes identified**		**Number of attributes in objective function**	
	Single	**Multiple**	**Single**	**Multiple**
Problems & Algorithms	- Generic shortest path algorithm - Label setting algorithm - Label correcting algorithm	- k-shortest path algorithm	- Traditional shortest path algorithm - k-shortest path algorithm	- Multi-Criteria shortest path

Another classification of the existing SPP and extensions could be based on the time-dependency of the link attributes. If the link cost changes with the time of day, identifying the shortest path is defined as a dynamic shortest path problem or shortest path problem in a dynamic traffic network. Lastly, in many transportation situations, the link travel time in a network is not deterministic but is a discrete or continuous stochastic process. That is, the cost or travel time on each link may be considered as a random variable. There exists a large amount of literature on the SPP in a dynamic and static network that require procedures to model those network characteristics.

The following section focuses on the review of the standard SPP algorithms that identify a single route based on a single attribute in a static, deterministic network.

2.1.1 Labeling Algorithms for the Shortest Path Problem

Labeling algorithms are the most popular and efficient algorithms for solving the SPP. These algorithms utilize a label for each node that corresponds to the tentative shortest path length pi to that node. The algorithm proceeds in a way so that these labels are improved until the shortest path is found. There are two types of labeling algorithms: label setting (LS) and label correcting (LC). The LS algorithm sets the label of one node permanently at each iteration, thus increasing the shortest path vector by one component at each step. The LC algorithm does not set any label permanently. All the components of the shortest path vector are obtained simultaneously, after the algorithm terminates. A predecessor label is stored for each node that will represent the previous node in the shortest path to the current node. This is used to construct the shortest paths to each node by backtracking. Table 2.2 gives a comparison of these two algorithms.

Table 2.2. Comparisons between label-setting and label-correcting algorithms

	Label-setting algorithm	**Label-correcting algorithm**
Applicable	- Shortest path problems defined on acyclic networks with arbitrary arc lengths. - Shortest path problems with nonnegative arc lengths.	- More general and applies to all classes of problem, including those with negative arc lengths. - Shortest path problems with arbitrary arc lengths.
Efficiency	- Much more efficient. - Much better worst-case complexity.	- More algorithmic flexibility.
Algorithms	- Dijkstra algorithm (Dijkstra, 1959).	- Bellman-Ford-Moore algorithm (Bellman, 1958; Moore, 1957; Ford, 1956). - Incremental-graph algorithm (Pape,1974 and Pallottino, 1984). - Threshold algorithm (Glover et al., 1984). - Topological ordering (Goldberg and Radzik, 1993).

2.2 K-Shortest Path Problem and Algorithms

In many transportation applications, there is a need to identify a multiple number of paths. Drivers may wish to make explicit trade-offs between routes, such as taking a longer route that has a lower variance or fewer stops. There is currently no algorithm that can determine the optimal route in this situation. A possible approach to both of these situations is to identify a number of distinct routes that then, using some multiple criteria decision making (MCDM) techniques, identify the best route. A common subset of the problem to identify a multiple number of routes is the k-shortest path problem (K-SPP)

Two classes of the k-shortest path problems in static networks have been investigated. In the first class, optimal paths are not allowed to contain loops. This class of problems was studied by several authors including Bellman (1958), Fox (1975, 1978), Lawler (1972, 1977), Minieka and Shier (1973), Perko (1986) and Yen (1971). In the second class, paths may contain repeated nodes. Authors who studied the second class of problems include Bellman (1958), Dreyfus (1969), Fox (1973), Hoffman and Pavley (1959), and Lawler (1972). Minieka and Shier (1973) and Shier (1976, 1979) appear to be the first who discovered and exploited algebraic structures that exist between the usual shortest path and the k-shortest path problems. The formulation of the k-shortest path problem in dynamic networks can be viewed as an adaptation of a static k-shortest problem formulation in the time-expanded equivalent network representation of a discrete-time dynamic network.

2.3 Multi-Criteria Shortest Path Problem

The previous two sections have provided reviews of the traditional shortest and K shortest path algorithms which are concerned with only one route attribute or travel time. However, drivers consider a number of criteria when selecting routes, and may have different preferences or utility functions when selecting a best route. It is therefore necessary to take into account various route attributes and the drivers' preferences when identifying an optimal route.

There is a rich source of literature on the multi-criteria shortest path problem (MCSPP) in the operations research and management science areas. The existing algorithms for the MCSPP may be classified into two groups. In general, the first group generates all non-dominated paths while the second group focuses on the problem of finding the optimal path based on the users' objectives.

The difficulty in solving the MCSPP may be attributed to the fact that there may be no single optimal solution (i.e., path in this dissertation) that satisfies all objectives simultaneously. If there were, the solution to the MCSPP would be very straightforward because the best path would dominate all other paths in terms of all objectives. Due to the non-existence of the overall best solution, a set of non-dominated paths or Pareto optimal paths, from which the decision maker must select the most preferred or most compromising solution, must be generated. The existing approaches for the MCSPP without utility function are broadly classified into two, as follows and they are summarized in Table 2.3.

Table 2.3. Existing approaches for the MCSPP without a utility function

Approach		Output	Advantages /disadvantages	Algorithms
Exact approach	Generate all non-dominated paths by - modified label setting or correcting algorithm - k-shortest path algorithms	multiple paths	• Exponentially increasing number of non-dominated paths (NP-hard) • Set contains optimal path	Hansen (1980) Martins (1984) Henig (1985) Corley and Moon(1985) etc.
Approximation	Estimate the non-dominated path set to some predetermined degree of accuracy using a scaling and rounding technique	multiple paths	• Enhancing computational efficiency • Set may not contain optimal path	Warburton (1987)
	Apply A* search technique	multiple paths		Stewart and White (1989)

A traditionally employed methodology for the MCSPP would be to generate the entire set of non-dominated paths. The exact algorithms for generating the entire set of Pareto optimal paths may be classified based on their methods: 1) the labeling method and 2) the k-shortest path algorithm or linear programming-based approach. Hansen (1980), Henig (1994), and Sancho (1988) extended generic label setting shortest path algorithms, such as Dijkstra's (1959), into a multiple-labeling scheme, while Loui (1983), Corley and Moon (1985), and Brumbaugh-Smith and Shier (1989) extended general label correcting algorithms such as Moore's (1957). Brumbaugh-Smith and Shier implemented the labeling correcting algorithm in an artificial two-attribute network with varying size and varying degree of correlation between the two artificial attributes. Bicriterion algorithms of Climaco and Martins (1982) and Henig (1985) are based on the k-shortest path algorithm. The algorithm first computes the fastest path and the cheapest path, and computes the *j*-th cheapest paths until the cost of the *j*-th cheapest path is the same as the cost of the fastest path.

Aside from the above mentioned generic MCSPP, there have been relatively few attempts to incorporate multiple criteria within route choice modeling for transportation problems.

The review by Current and Marsh (1993) describes the various approaches quite well. Multiobjective routing of hazardous material shipments is an important application of such methods. The reviews by List et al. (1991) and Erkut and Verter (1995) provide insight into hazmat applications, and more recent hazmat routing and scheduling efforts (e.g., Nozick, List, and Turnquist 1997; Miller-Hooks and Mahmassani 1998b) have begun to merge multiobjective routing with time-dependent and stochastic attributes.

Dial (1996) formulated a bicriterion user equilibrium assignment model based on out of pocket costs and trip time based on the previous study which assumes a linear utility function (i.e., weighting method). Blue et al. (1997) proposed an algorithm for the MCSPP which considers two attributes: travel time and route complexity. The route complexity is represented by turning maneuvers. The algorithm is based on the simple weighting method and assumes that all members of a particular user class use the same value of weight, under the assumption that the nonlinear utility function is known a priori. Scott and Bernstein (1998)'s algorithm generates a set of Pareto optimal paths using a CSPP and then identifies the best path by evaluating the utility values of the alternative paths. It should be stressed that none of the existing MCSPP algorithms discussed above are concerned with the route similarity in terms of links used, which is a critical aspect for alternative paths from the drivers' point of view.

The generalization of the stochastic dynamic shortest path problem to multiple objectives, creating the multiobjective stochastic dynamic shortest-path (MSDSP) problem, results in a problem that is very difficult to solve. To our knowledge, only two studies, Turnquist (1987) and Miller-Hooks and Mahmassani (1998b), have been done on the MSDSP problem. Both of these studies focus on applications to hazmat transportation.

2.4 Stochastic Shortest Path Problem

In many transportation applications of the SPP, the travel time of each link is not really fixed but is, in fact, a random variable. The problem of determining the optimal path in this type of network is known as the stochastic shortest path problem (SSPP). The uncertainty of the link travel times is an important factor to be modeled in ITS because (1) there is an inherent uncertainty or randomness in link travel times, (2) the route optimization is based on the link travel times forecasted over the multiple periods into the future and, accordingly, the uncertainty is expected to increase as link arrival time increases, and (3) travel time reliability or variance of a route is one of the crucial criteria for route choice.

If the underlying network is assumed to be static (non-time-dependent), the link travel times remain unchanged after they are revealed to the travelers. In a time-dependent network, on the other hand, the travel time of every link at every time period is an individual random variable, so travel times revealed at different time periods could be different. The study of SSP problems in static networks is useful to the study of its time-dependent counterpart.

Different types of stochastic shortest path problems have been considered with a different meaning for the optimal path. One of the most considered criterion for determining the optimal path is one that maximizes the decision maker's expected utility. Such a criterion stems from the Von Neumann- Morgenstern formulation of how preference judgments are made under uncertainty (Loui, 1983). Another good definition of an optimal path is one that maximizes the probability that its length does not exceed a pre-specified threshold value (Frank, 1969). The same criterion has also been used by Henig (1990) for the stochastic knapsack problem. A closely related stochastic shortest path problem involves chance constraints. An optimal path minimizes the threshold value, while satisfying the constraint that the probability of the path length exceeding this threshold value is at most, a pre-specified value α. Such a criteria was considered by Henig (1990) for the knapsack problem, and by Ishii et al. (1981) for the minimal spanning tree problem.

One possible way for computing the distribution of shortest length is by formulating the problem as a stochastic linear program with random objective coefficients. Bereanu (1966) and Eubank (1974) proposed methods for computing the distribution of the optimal objective value when the coefficients are continuous random variables. These methods require the evaluation of the probability that a given basis is optimal. This is a task that requires a complicated partition of the state space of the objective function. Frank (1969) and Sigal et al. (1980) presented exact methods, both of which rely on the evaluation of multiple integrals. Because of the great complications that arise in those evaluations, they suggested Monte-Carlo sampling. Kulkarni (1986) presented an analytic method for the exact computation of the distribution of shortest distance. It is based on a Markov process with an absorbing state when the arc lengths are independent and on exponentially distributed random variables. He constructed a continuous time Markov chain with a single absorbing state from the original network. Time until absorption in this absorbing state starting from the initial state is equal to the length of the shortest path original network. Algorithms are also developed for computing the probability that a given path is the shortest path in the network. This is for computing the conditional distribution of the length of a path given that it is the shortest path in the network.

The issue of arcs being random plays an important role especially in communication networks. Two computational tools are often important in the design and analysis of the communication networks. One is the method to compute network reliability and the other is the method to compute the response time of the system. By reliability, it is meant the probability of connectivity between a given source and sink node. By time response, it is meant the expected time delay that a message that originates at the source node sustains before it reaches a sink node. Hansler (1972) has proposed an interesting algorithm on the reliability of networks based on generating mutually exclusive cut sets and calculating probabilities of related events. In the literature dealing with reliability of networks (and also in the literature related to switching circuits, communication networks, and traffic networks), there are many methods that deal with various algorithms dealing with reliability of networks.

Mircandi (1976) analyzed the calculation of reliability of various emergency networks. His approach starts with sorting all (*s, t*) paths and creating a disjoint expression by comparing neighboring paths. This expression is then used for computing the probability that there exists an (*s, t*) path of length less than or equal to a fixed value or the mean of the shortest path length.

Sigal et al.(1980) introduced the concept of path optimality index as a performance measure for selecting a path in a stochastic network. A path optimality index is defined as the probability of a given path being shorter than all other network paths. Uniformly directed cutsets are introduced by them. It was further studied by Adlakha (1986) and Kamburowski (1985). Also, Alexopoulos (1997), and Seok and Pulat (2000) generated the probability distribution function of path travel times using probability axioms, Markov Chains, or simulation and then selected the path with the highest probability of being the shortest.

The theory of analyzing the stochastic shortest path problems, using the Markovian decision problems, applies only when the arc costs are non negative or all non positive. The deterministic theory of shortest path problems allows arc lengths that can be negative, as well as positive. Bertsekas and Tsitsiklis (1991) provided an analysis of the stochastic path problems that generalizes the known results of the deterministic counterpart.

A factoring approach for the stochastic shortest path problem was suggested by Hayhurst and Shier (1991). In their work, the authors assumed that the arc lengths are discrete random variables assuming a finite number of non-negative integer values. Also, the arc lengths are assumed to be statistically independent. Since the arc lengths assume random length, the length of the shortest path is a random variable and the authors were interested in finding the distribution of the arc lengths. The factoring approach is based on the concept of structural factoring, in which a stochastic network is decomposed into an equivalent set of smaller, generally less complex sub networks. Several networks are identified and exploited to significantly reduce the computational effort required to solve a problem relative to complete enumeration. This algorithm can be applied to two important classes of stochastic network problems. One is determining the critical path length distribution for acyclic networks and the other, terminal reliability for probabilistic networks.

Bard and Bennett (1991) developed heuristic methods involving Monte-Carlo simulation to solve the stochastic shortest path problem with a general non-increasing utility function. They showed that their heuristic was able to solve a large number of randomly generated test problems, with sizes ranging from 20 to 60 nodes.

Corea and Kulkarni (1993) proposed a methodology for computing the distribution of shortest length and criticality indices of paths. They assumed that the arc lengths are integer-valued, replaced each arc with largest possible length *m* by a sub-network with *2m* arcs, and constructed Markovchains with absorbing state and binary transition costs. The above measures are computed by evaluating the distribution of the total cost incurred until absorption. Unfortunately, their construction limits the applicability of the methods to problems of small size.

Cai, Kloks, and Wong (1996) studied the time varying shortest path problems with constraints. They studied the problem in which the objective is to study the shortest path subject with the constraint that the total traverse time is at most, some number T. In this study, the authors addressed the situation where the transit time and the cost to traverse an arc that is varying over time, and depending upon the departure time at the beginning vertex of the arc. Waiting times at vertices are considered decision variables. The problem is to find an optimal path as well as the optimal waiting times at the vertices along the path, subject to the constraint that the total traverse time of the path is, at most, T. The authors tested the model for a variety of applications.

The literature shows a combination of stochastic network concepts with interesting concepts of the utility functions. Murthy and Sarkar (1997) considered a stochastic shortest path problem of determining a path that maximizes the expected utility. The nature of the utility functions used to evaluate paths was of decreasing deadline type. Algorithms based on pruning techniques were developed for this case. One of the two algorithms makes use of the concept of local preference relations while the other type makes use of the relaxations.

State Space Partitioning methods, developed by Alexopoulas (1997), examine discrete arc random lengths. The method is used for developing computing measures for shortest paths. The computation measures include the probability that a path exists where the length doesn't exceed a specified value and the probability that a given path is shortest. These methods are based on an iterative partition of the network space and provide bounds that improve after each iteration and eventually become equal to the respective measure. These bounds can also be used for constructing simple variance reducing Monte Carlo sampling plans, thus making the algorithm useful for large problems. The algorithms can be easily modified to compute performance characteristics in stochastic activity networks.

Andreatta and Romeo (1988) study the problem in a static network where the topology is stochastic. A stochastic topology is defined by a deterministic set of nodes N and a random set of links. Each possible topology has a positive probability. A random link can be either active or not. When it is active, it is included in the network; when it is not active, it is removed from the network. The decision maker (DM) can learn whether a link is active or not once he/she reaches the node from which the link emanates. The DM can reroute once he/she finds out the next link is inactive. They prove four facts about a stochastic shortest path that are different from those about a deterministic shortest path. A stochastic dynamic programming formulation of the problem is provided, with the definition of "information state" which reveals the active/inactive links of the network to the decision maker so far and based on which, the recourse decision is made. It is pointed out that the complexity of the algorithm can grow exponentially with the number of links.

Polychronopoulos and Tsitsiklis (1996) extend the work of Andreatta and Romeo (1988). They study the problem both in networks with link travel times that are correlated and in networks with independent link travel times. For the dependent case, a joint distribution of link travel times is used to represent the stochastic network. We can see that the stochastic topology in Andreatta and Romeo (1988) is actually one special form of joint distribution of link travel times. It is assumed that the travel time realizations of outgoing links of a given node are known and remembered by the traveler once he/she arrives at this node, and the realizations remain unchanged afterwards. As the traveler moves on the network from the origin to the destination, more link travel time realizations are learned, and the network becomes closer to a deterministic one. The concept of an information set is introduced to represent the traveler's knowledge about the network. An information set is composed of support points that are consistent with the link travel times revealed so far. When the information set becomes a singleton, the network becomes deterministic. A similar approach is designed for the independent case, with changes in the manner in which the information set is defined. The algorithms, however, have exponential running times: the algorithm for the dependent case has a running time exponential in the number of support points, and the algorithm for the independent case exponential in the number of links. It is proved that the problem with dependent link travel times is NP-complete, and that with independent link travel times is NP-hard. Some heuristics are given and the relationships between results from heuristics and exact algorithms are studied.

Cheung (1998) studies the problem with the same independent network assumptions as those in Polychronopoulos and Tsitsiklis (1996), except with the assumption that two visits to the same node result in two independent realizations of outgoing link travel times. This assumption (which is termed as "reset" later by Provan (2003)) actually makes ambiguous the statement that the network is static, as the same link can take different travel times at different times, although the distribution is the same. On the other hand, the reset assumption makes possible a simple recursive equation for the expected minimum travel times. An approach that mimics the classical label-correcting algorithm is presented. Computational tests are carried out to compare different implementations of the label-correcting approach. Provan (2003) studies the same problem as defined by Cheung (1998) with the extension that the link travel times can be dependent. However, this relaxation from independent to dependent networks does not make the problem harder. In fact, the reset assumption makes the term "dependent" less clear, as one can never make inferences about travel times on links other than those going out of the current node. The same recursive equation is presented, but a polynomial-time algorithm is designed and its complexity analyzed.

The shortest path algorithms also have been found to be applicable to compute shortest paths in time-dependent (but not stochastic) networks (Dreyfus, 1969; Orda and Rom, 1990; Kaufman and Smith, 1993; Ziliaskopoulos and Mahmassani, 1993; Chabini, 1997 and 1998).

Selection of minimum variance paths is studied by Frieze and Grimett (1985). Sen et al (2001) present a parametric 0-1 quadratic programming approach to select a path with the least mean-variance. A variance-constrained shortest path problem for hazardous material transport had been dealt with earlier by Sivakumar and Batta (1994). Sen et al (2001) proposed a mean-variance model for route selection assuming time independence. They proposed a bicriteria network flow model where the objectives are to minimize expected travel time (linear) and minimize variance of the total trip time (quadratic). The variance-covariance matrix is assumed to be positive definite to avoid cycles during the path selection process. A parametric approach is used to determine the set of efficient solutions. For each parameter value, the optimal solution to the continuous relaxation of the problem is determined. If the solution contains multiple routes, the route with the least objective function value is selected. The method is not difficult to implement and includes link dependencies.

2.5 Stochastic Time-Dependent Shortest Path Problems

In the stochastic time-dependent shortest path problem (STDSPP), the link travel times are time varying random variables and are modeled using probability density functions and time-dependency.

Hall (1986) proposed an approach combining branch-and-bound and k-shortest paths techniques for determining the least expected time path in a stochastic time-dependent network where the path is chosen a priori. The algorithm required that the expected times and least possible times be calculated for each path; however, no procedure was given for calculating these values. There is no guarantee that the algorithm will terminate before all paths have been evaluated. A heuristic method was suggested in Kaufman and Smith (1993) for improving the computational time of this procedure. They generalized the rules for the use of LS algorithms for TDSP problems by including the stochastic case. They prescribed the use of expected values instead of deterministic values and showed that under the consistency assumption, one may obtain results similar to the deterministic case.

Psaraftis and Tsitsiklis (1993) considered optimal policies for determining the least expected cost path between an origin and destination in an acyclic, dynamic and stochastic network. The cost of traveling on arcs, leaving each node, is associated with a finite-state Markov process, which varies randomly, but independently, of the states of the other nodes of the network and is known only upon arrival. Waiting is permitted. Koutsopoulos and Xu (1993) have shown that time-dependent link delays can be modeled as a Markov process.

Fu and Rilett (1998) conducted the first study which explicitly estimated route mean travel time and variance based on link information typically available in transportation networks. Using a Taylor series expansion, they proposed first and second order route mean travel time and variance approximation algorithms. Subsequently, they developed a heuristic approach to determine the expected shortest path. Instead of enumerating all the possible paths, the heuristic algorithm generates multiple numbers of paths using traditional k shortest path algorithm, and identifies a path with a minimum expected route travel time among them. The tradeoff between the number of paths considered and the probability of finding the optimal solution were analyzed.

Miller-Hooks and Mahmassani (1998) proposed two efficient procedures to determine the least possible travel time paths from all origins to a single destination in networks where the link travel times are independent, discrete, time-dependent random variables that are permitted to operate under non-FIFO conditions. The first algorithm determines the least possible time path from each node to a destination node for each departure time in the time period and a lower bound on the associated probability of the occurrence of this travel time. The second algorithm determines up to k least possible time paths, the associated travel times, and the corresponding probability of occurrence of the travel times. Both algorithms are an extension of the label correcting-based SPP. The authors proposed several algorithms for determining a priori paths in STD networks that employ such path-comparison techniques. In a subsequent study, Miller-Hooks and Mahmassani (2000) investigated the all-to-one variant of the problem. They presented two specialized modified label correcting algorithms for the problem of generating least expected time paths in stochastic time dependent networks. First, the expected value algorithm was presented for generating all a priori least expected time paths with associated expected times from all origins to a single destination for each departure time. Second, the expected lower bound algorithm was presented as an efficient procedure for determining lower bounds on the expected times of the least expected time paths without any associated path information. Miller-Hooks (2001) presented a specialized label-setting algorithm, the stochastic decreasing order of time algorithm, for determining the adaptive least expected time hyperpaths in stochastic time dependent networks. The author compared the performance of both label-correcting (expected lower bound) algorithms and label-setting algorithms. The results showed that the expected lower bound algorithm performed better on average than predicted by worst-case complexity.

Chabini (2001) developed an efficient solution algorithm based on the concept of the decreasing order of time for stochastic networks. This algorithm extends the decreasing order of time algorithm developed by Chabini (1997). His algorithm was shown to be computationally efficient both in theory and in practice. Gao and Chabini (2001) specified the best routing policy based on the availability of information access defining which arc travel time realizations are available to the travelers at any given time and node. They performed four different approximations techniques: (1) the certainty equivalent approximation, (2) the no-information approximation, (3) the open loop feedback certainty equivalent approximation, and (4) the open loop feedback with no-information approximation. There was a trade-off between effectiveness and efficiency for all approximations. They could have satisfactory running times, but their results could be arbitrarily worse in absolute value than those obtained by running the exact algorithm. The computational tests studied the relationship between some parameters and the performance of approximations.

Chapter 3. Background and Framework

In deterministic networks, the least time path is defined simply and explicitly. However the nature and complexity of the least time path is different for a stochastic network. In this chapter, several key concepts of stochastic, time-dependent networks are described. These concepts are critical to the development of the algorithms presented in Chapter 4 and 5. The first two sections of this chapter, notations and some of the concepts of stochastic time-dependent networks are described. In section 3.3, the basis for selecting one path over another, when the path's travel times are random variables with probability distribution functions that vary with time is discussed.

3.1 Notation for Stochastic Time dependent Networks

Let G = (N, A, T, TI, P) be a directed graph where N is the set of nodes, $|N|=n$, and A is the set of arcs, $|A|=m$. It is assumed that the travel times along the arcs are represented by discrete random variables whose distribution functions are time-dependent during the period of interest, $t_0 < t < t_0 + (I)\delta$, referred to as the "peak period", and are stationary any time thereafter, $t > t_0 + (I)\delta$. This formulation can be generalized to travel times with continuous distributions. The network is considered at a set T of discrete times $\{t_0 + n\delta\}$, where n is an integer, $n = 0, 1, \ldots, I,$ and δ is the smallest increment of time over which a perceptible change in the travel time distributions will occur for $t \in T$.

For each departure time $t \in T$ and each arc $(i, j) \in A$, the set $TI(t)$ of non-negative real valued possible travel times $\tau_{ij}^{k}(t)$ for traversing the arc at a given time t is given, $k=l, \ldots, K_{ij}(t)$, where $K_{ij}(t)$ is the number of distinct travel time values on arc (i, j) possible at time t. Travel time $\tau_{ij}^{k}(t)$ occurs with the probability $p_{ij}^{k}(t)$, where $p_{ij}^{k}(t) \in P(t)$ t and

$$\sum_{k=1}^{K_{ij}(t)} p_{ij}^{k}(t) = 1, \qquad \forall\ t \in T$$

It is assumed that $\tau_{ij}^{k}(t) = \tau_{ij}^{k}(t_0 + I\delta)$ and $p_{ij}^{k}(t) = p_{ij}^{k}(t_0 + I\delta)\ \forall\ k = 1, \ldots, K_{ij}(t)$ and $(i,j) \in A$ for all t occurring after the peak period, i.e. $\forall\ t > t_0 + I\delta$. The set of travel times and the corresponding set of probabilities with which each travel time will occur, (TI, P), are assumed to be given.

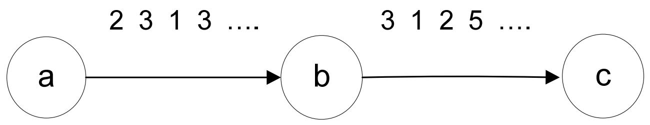
The arc travel time probability distribution functions are assumed to be independent across arcs and over time and no waiting is permitted at any intermediate node. The network is assumed to be non-FIFO. Such a network is referred to as a stochastic, time-dependent network. This stochastic, time-dependent network definition is an expansion of the deterministic, time-dependent network described by Ziliaskopoulos and Mahmassani (1993).

This dissertation addresses the problem of determining "preferred" paths $\forall i \in N$ to a given destination, $\boldsymbol{d}$, for each $t \in T$ in stochastic, time-dependent networks. While this problem has some of the same elements as the problem of determining least time paths in deterministic, time-dependent networks, where the arc travel times change dynamically over time, but occur with probability-one in a given time interval, the added dimension of stochasticity dramatically increases the difficulty of the problem.

3.2 Definitions for Stochastic, Time-Dependent Networks

3.2.1 The Space-Time Expansion

It is common to represent dynamic problems by space-time networks (Powell, Jaillet and Odoni, 1995). Consider a street network where the arc travel times are deterministic, dynamic quantities. If this network is graphically represented without incorporating time as a dimension, the graph, G. will consist of a set of nodes that represent intersections, and a set of arcs, that represent the streets. A vector of travel times (or some measure of cost) is associated with every link. This vector represents travel time for the given departure time. An example network is shown in Figure 3.1 where the arc travel times are given by row vectors and time moves from 0, increasing to the right in constant increments of time. It is assumed that the travel times are given in the units of these time increments.



Time (t)	0	1	2	3
Arc a-b	2	3	1	3
Arc b-c	3	1	2	5

Figure 3.1. Deterministic time-dependent network G

This network is expanded to a space-time representation G', where time increases from left to right and each node of G' corresponds to a node in G at a given departure time. This graphical representation can be extended for use in dynamic networks with stochastic arc weights. Since there may be more than one possible travel time for each departure time from a node in a stochastic network, several arcs may originate from each space-time node. However a deterministic network can have only one arc originating from each space time node. If the arc weights are continuous random variables, then an infinite number of arcs may emanate from each space-time node.

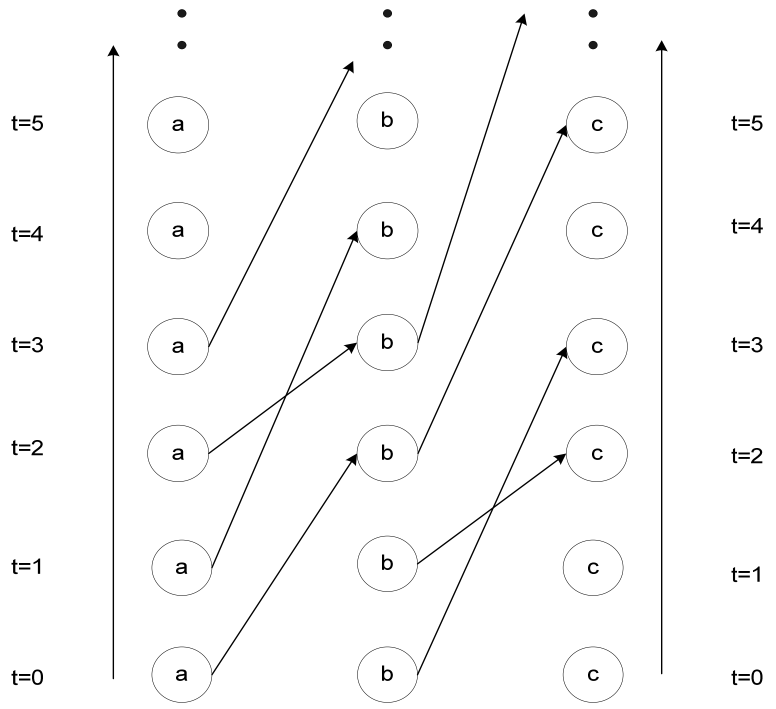
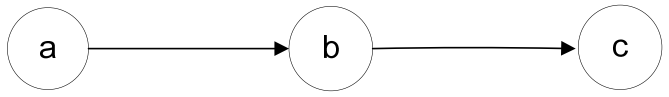


Figure 3.2. Space-time expansion of G: G'



	Arc a-b		Arc b-c		
Time (t)	0	1	0	1	2
	1 (0.4)	1 (1.0)	1 (0.5)	2 (0.3)	2 (0.1)
	2 (0.6)		3 (0.5)	4 (0.7)	3 (0.5)
					4 (0.4)

Figure 3.3. Stochastic, time-dependent network E

Consider the network of Figure 3.3, with discrete random arc times whose probability mass functions (PMF) vary with time. For each arc, for the given departure time *t*, the travel times with positive probability are given. For example, for departure time 1 from node **b**, arc (b, c) will have a travel time of two units with the probability 0.3. In the figure, the PMF's for each arc are shown at only a few time intervals, for clarity. The space-time expansion of this network is shown in Figure 3.4, where the arcs have weights corresponding to the associated probability of occurrence.

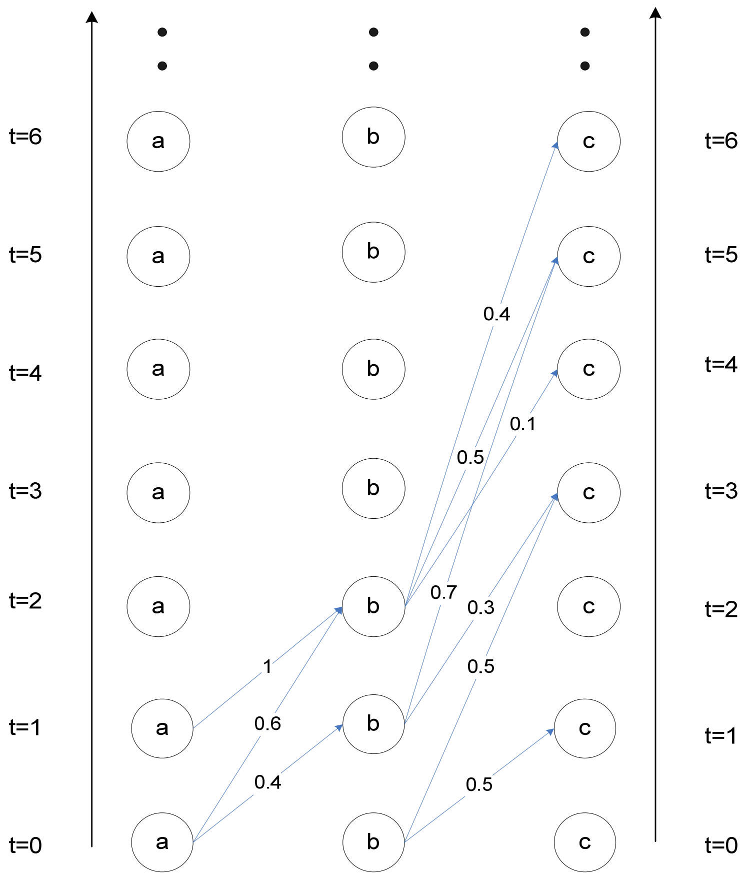


Figure 3.4. Time-Space network of E: E'

If the arc travel times are independent, the probability of each arrival time at node **c** is determined from the multiplication of the probabilities associated with the appropriate arcs of E'.

For example, the probability of arriving at node **c** at time 3, if one leaves from node **a** at time 0, is 0.4*0.3=0.12. Similarly, the probability of arriving at node **c** at time 4, if one leaves from node **a** at time 0, is 0.6*0.1=0.06. In the same way, the probability of arriving at node **c** at time 5, if one leaves from node **a** at time 0 is (0.28 + 0.30) = 0.58. The network in Figure 3.3 is trivial in that only one topological path exists between nodes **a** and **c**; however, if more than one path were possible, the path with the least possible travel time could be identified directly from the graph, E'.

3.2.2 The Parameters for Path Selection

The selection of a single "best compromise" path from the set of non-dominated least time paths ultimately depends on the decision-maker's preferences, and is likely to be situation-specific. For instance, a risk-averse decision-maker may choose the path that has the smallest probability of being the longest. On the other hand, a risk seeker may choose the path with the highest probability of being the shortest. Typically, the decision-maker considers certain trade-offs, between expected travel time and variance. In opting to get the better in travel time a decision maker has to compromise in variance. Several measures can be considered for selecting the best-compromise solution for a given situation. The list hereafter is by no means all inclusive, and is intended to present some examples of the logic and preferential basis that a decision-maker employs in order to select one path over another.

Let $L_i^h(t)$ be the travel time random variable of the hth non-dominated path from node *i* to the destination for departure time *t*, with the cumulative distribution function F($L_i^h(t)$). Let p_i^t be the set of non-dominated paths from the node at departure time t. The following measures may be considered individually or jointly in the selection among non-dominated paths:

<u>*Least Expected Value*</u>

Select the path for which E[$L_i^h(t)$], $h \in p_i^t$, is minimized. Although the expected value is a commonly used measure, it may not be the most appropriate for many applications. If the application involves a single or only a few trials, other measures may provide more appropriate criteria for path selection.

<u>*Least Variance*</u>

Variance is often used as a replacement for risk: Var[$L_i^h(t)$], $h \in p_i^t$.

<u>*Smallest probability of being longest*</u>

Selecting the path with the smallest probability of being longest may be appropriate for the risk-averse decision-makers. Choose the path $h \in p_i^t$ that minimizes $\Pr\{L_i^h(t) \geq L_i^g(t),\ \forall g \neq h \in p_i^t\}$. Note that this measure does not preclude the possibility that the path may be very long.

<u>*Largest probability of being shortest*</u>

The truck dispatcher may wish to select the path $h \in p_i^t$ that maximizes $\Pr\{L_i^h(t) < L_i^g(t),\ \forall g \neq h \in p_i^t\}$. Again, such a path may have some probability of being very long. These measures can be used as criteria to select the best-compromise path.

3.2.3 Expectation and Paths

The expected value criterion is of particular interest because it can be used to reduce stochastic to a deterministic problem. This can eliminate many complexities associated with comparing random variables. Although the expected value is a commonly used measure, it may not be appropriate in all situations. If the probability that a path will take a certain length is interpreted as a long-run relative frequency, then the expected value for the random variable is defined for an infinitely large number of repetitions (Kalbfleisch, 1985). That is, if a vehicle travels over the path with the least expected travel time many times, then in the long run, the average travel time over all the trips will be shorter than had another path been selected for repeated traversal. Depending on the application, this long-run minimum path may or may not be appropriate. If the application involves a single or only a few trials, other measures may provide more appropriate criteria for path selection.

A closely related measure with very different implications is the least expected travel time through a stochastic network. This can be determined by computing the sum of the arc costs on the least costs path for every realization of the network state weighted by the probability of such a realization. A similar method can be used to determine the pmf of the minimum travel time. This is most useful if one is interested in finding the expected arrival time, or the pmf of the minimum travel time, of a vehicle in a transportation network, or a packet in a communication network, to a destination node given that the shortest path at the time of departure will be selected. In a reliability framework, Mirchandani (1976) shows that the original stochastic network can be transformed to an "emergency equivalent network" from which the expected least travel time through the network can be computed. See (Hagstrom, 1990) for related work on this problem.

The following example illustrates the difference between determining the least expected travel time through a network and the expected time of the least expected travel time path.

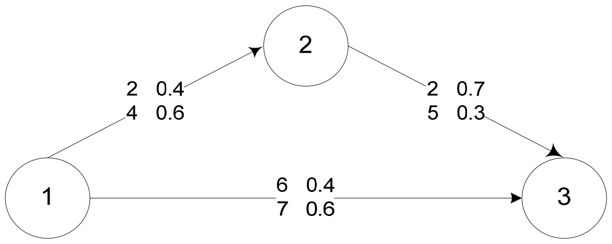


Figure 3.5. A stochastic network

From Table 3.1, the probability mass function of the minimum travel time between nodes 1 and 3 of the network in Figure 3.5 is (4, 6, 7) with the corresponding probability of occurrence of (0.28, 0.54, 0.18), respectively. The expected least travel time from node 1 to node 3 is 5.62 units of time, determined directly from the pmf of the minimum travel time. If the arc travel time random variable is set to its expected value, the expected travel time on the least expected time path is 6.1 units of time, as shown in Figure 3.6:

Table 3.1. Determining expected least time through network of Figure 3.5

Arc (1,2)		**Arc (2,3)**		**Path 1-2-3**	**Arc (1,3)**		**Least Path**	
Travel Time	Prob	Travel Time	Prob	Travel Time	Travel Time	Prob	Travel Time	Prob of Realization
2	0.4	2	0.7	4	6	0.4	**4**	0.112
4	0.6	2	0.7	6	6	0.4	**6**	0.168
2	0.4	5	0.3	7	6	0.4	**6**	0.048
4	0.6	5	0.3	9	6	0.4	**6**	0.072
2	0.4	2	0.7	4	7	0,6	**4**	0.170
4	0.6	2	0.7	6	7	0.6	**6**	0.252
2	0.4	5	0.3	7	7	0.6	**7**	0.072
4	0.6	5	0.3	9	7	0.6	**7**	0.108
		Expected least travel time = 5.62						

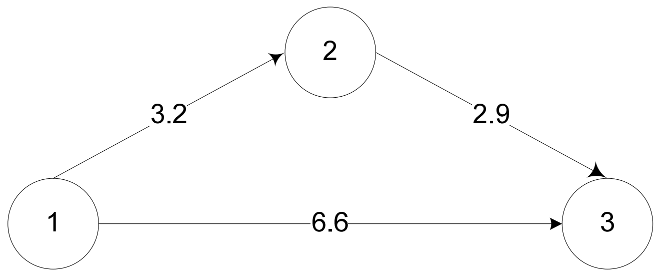


Figure 3.6. Expected arc travel time

The expected time of the least expected time path provides an upper bound on the expected least travel time through the network. Likewise, the expected least travel time through the network is a lower bound on the expected travel time of the least expected time path. The expected least travel time through the network is not necessarily the expected travel time on any particular path, nor is it necessarily a feasible travel time on any path.

For certain applications, such as routing messages between nodes in a communication network, the pmf or expectation of the minimum time between the two nodes of a network may be required. However, neither measure provides path information as these times come from the composition of more than one path. Some applications, on the other hand, require actual path information and thus, these bounds would be insufficient.

In a network where the arc times are random variables with time-independent pmfs, one can simply set the random travel times to their expected values and apply a deterministic shortest path algorithm to determine the path with the least expected travel time, i.e., the path with lowest sum of constituent expected arc times, as shown in the previous example. However, in a time-dependent, stochastic network, the least expected time (cost) path can no longer be determined by setting each arc time to its expected travel time and solving the equivalent deterministic problem {Hall, 1986), because the travel time on an arc now depends on the time of arrival at its origin node. This is further explained in Proposition 3.1.

Proposition 3.1. In a network with random arc travel times with time varying pmf's, the least expected time path cannot be determined by setting each arc time random variable to its expected value and solving an equivalent deterministic, time-dependent problem.

Proof. Assume that time can be discretized into small time intervals. Within each time interval, the pmf's of the arc travel time random variables are assumed to be constant and the travel times are given in multiples of these time intervals. The proof proceeds by counter example.

Suppose a network is given with discrete probability mass functions of the travel limes at departure times, 5, as shown in Figure 3.7. For departure time 0, the expected travel time on path a-b-c can be calculated.

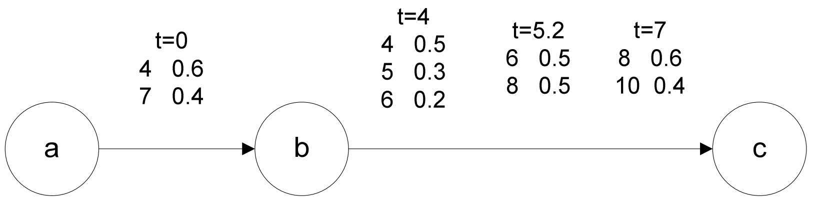


Figure 3.7. Example network with time-dependent pmf

Assume that each arc travel time random variable can be replaced by its expected value. Then the expected travel time on arcs ***a-b*** and ***b-c*** is 5.2 and 7 units of time, respectively, and the expected path length is 12.2 time units.

Instead of replacing each arc travel time random variable with its expected value, the expectation can be calculated directly. Thus, a vehicle departing node **a** at time zero can arrive at node ***b*** at either time 4 or time 7. Assuming that no waiting is permitted at node ***b***, the vehicle must leave immediately upon arrival. If arrival at node ***b*** is at time 4 then the travel time on path ***a-b***-c is 8 units of time with the probability 0.30 or 9 with the probability 0.18, or 10 with the probability 0.12. If arrival at node ***b*** is at time 7, then the path length for ***a-b-c*** is 15 and 17 units with probabilities 0.24 and 0.16 respectively. Thus the expected travel time on path ***a-b-c***, given the departure time from node ***a*** at time zero, is:

$$(8 \times 0.30)+ (9 \times 0.18)+ (10 \times 0.12)+ (15 \times 0.24)+(17 \times 0.16)= 11.54$$

And hence, there is no guarantee that the expected travel time on a path in a stochastic, time-dependent network can be calculated by setting each arc travel time random variable to its expected value and solving an equivalent deterministic, time-dependent problem.

3.3 Path Comparisons for Stochastic, Time-Dependent Networks

The concepts and methodology required for this comparison depend on the decision process along the path and information availability. If the entire path is specified before traveling begins, and no deviations en route are permitted, the non-dominated paths are selected a priori on the basis of only the time-varying probability distribution functions of the arc travel times. This is referred to as a priori path selection. Paths with lower actual travel times may be determined by allowing decisions at intermediate nodes, given that the additional information on the actual (revealed) arrival times is given at the intermediate nodes. Some of the paths determined a priori may never be "best" in this context, and therefore could be eliminated. Strict-comparison rules are presented for determining non-dominated paths where such decisions at intermediate nodes are permitted.

3.3.1 Criteria for A Priori Path Comparisons

Three criteria for comparing two paths at a single time interval are explored in this section: deterministic dominance, stochastic dominance and comparison via expected value. First consider two non-overlapping paths (paths that do not topologically share any arcs) between a given origin and destination at a given departure time. Because it is assumed that arc travel times are independent, the paths' travel times are independent. Let the two random variables x_1 and x_2, with distribution functions $F_{x_1}(t)$ and $F_{x_2}(t)$, denote the respective travel times on the two paths for the given departure time t. Comparing these paths is similar to comparing the two distribution functions $F_{x_1}(t)$ and $F_{x_2}(t)$. If the travel time along one path is at least as short as the other path for all possible realizations of the two paths and is shorter than the other for at least one realization, then this path is said to exhibit deterministic dominance (see Figures 3.8.a and 3.8.b where Path 1 deterministically dominates Path 2) for that time interval. If, on the other hand, neither path deterministically dominates the other path for this time interval, i.e. each path has some probability of being shorter than the other, then both paths are non-dominated, or efficient for this time interval. Likewise, a path that is deterministically dominated is called non-optimal, dominated or inefficient.

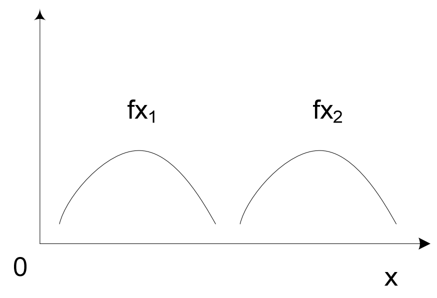


Figure 3.8.a. Path 1 deterministically dominates Path 2 at timt as seen by the non-overlapping density

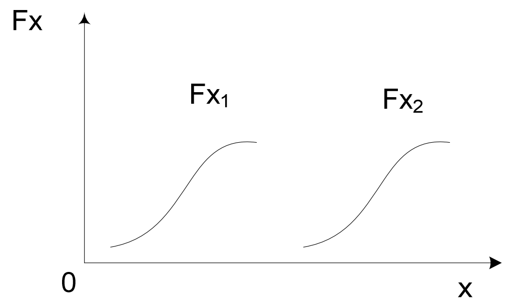


Figure 3.8.a. Path 1 deterministically dominates Path 2 at time t as seen by the distribution functions

Consider now two paths (between the same origin and destination) that topologically share one or more arcs. In a time-invariant network, the path travel times are no longer independent of one another. In a time-varying network, travel time (for the same departure time from the origin) on paths that topologically share an arc may still be independent if the arcs are not used at the same time intervals (under our assumption of the independence of arc travel times across time intervals).

It is incorrect to directly compare the distribution functions of two over-lapping paths via deterministic dominance because certain joint realizations of the respective path travel times may be impossible as they would imply different travel time values to hold simultaneously on the same shared arc(s). The following is a simple example in the stochastic network shown in Figure 3.9.

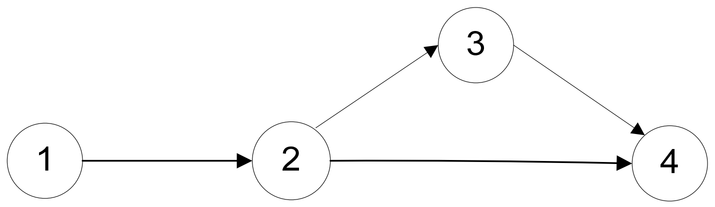


Figure 3.9. The over-lapping paths 1-2-3-4 and 1-2-4 share arc (1, 2)

Consider the probability mass functions for arc (1,2) and for subpaths 2-3-4 and 2-4 given in Table 3.2.

Table 3.2. PMFs of arcs and subpaths in Figure 3.9

Travel time (probability)		
Arc 1-2	**Subpath 2-3-4**	**Subpath 2-4**
2 (0.5)	2 (0.2)	6 (0.5)
5 (0.5)	3 (0.5)	7 (0.3)
	5 (0.3)	8 (0.2)

From the pmfs of the subpaths 2-3-4 and 2-4, any realization of subpath 2-3-4 is better than subpath 2-4. For any realization of arc (1,2), path 1-2-3-4 dominates path 1-2-4. Now consider the marginal (unconditional) pmfs of paths 1-2-3-4 and 1-2-4, given in Table 3.3.

Table 3.3. Unconditional pmfs of paths in Figure 3.9

Travel time (probability)	
Path 1-2-3-4	**Path 1-2-4**
4 (0.10)	8 (0.25)
5 (0.25)	9 (0.15)
7 (0.25)	10 (0.10)
8 (0.25)	11 (0.25)
10 (0.15)	12 (0.15)
	13 (0.10)

From the pmfs of the two paths, it appears that neither path dominates the other. Because travel time on path 1-2-4 could be 8 minutes, and path 1-2-3-4 could have a travel time of 8 minutes, it may be incorrectly concluded that there is some probability that path 1-2-4 will be better than path 1-2-3-4. In fact, there is zero probability of a joint realization of the respective path travel times where path 1-2-4 is better than 1-2-3-4. Path 1-2-4 can take 9 minutes only if arc (1, 2) takes 2 minutes, while path 1-2-3-4 can take 10 minutes only if arc (1,2) takes 8 minutes. A joint realization where path 1-2-4 is better than path 1-2-3-4 would require the travel time on arc (1, 2) to assume a value of 2 minutes and 5 minutes simultaneously, an impossible event under the assumptions of this problem. Therefore, it is not sufficient to simply compare the marginal distribution functions of two paths that share arcs in order to determine if deterministic dominance exists

Graphically, in Figure 3.10, Path 1 stochastically dominates Path 2 for a given time interval.

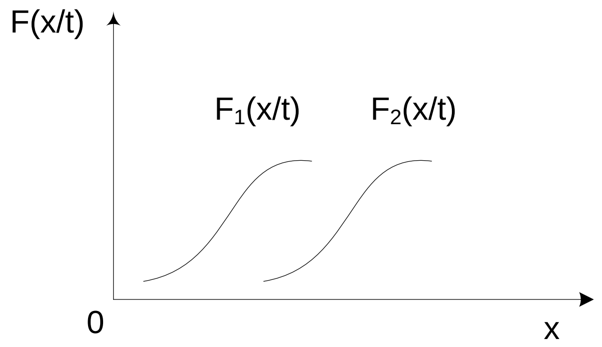


Figure 3.10. Path 1 stochastically dominates Path 2 at time t as seen by the non-intersecting distribution functions

Unlike deterministic dominance, stochastic dominance is established using full information from the distribution functions of the two paths. Even if the paths share one or more arcs, stochastic dominance between two paths can be established without conditioning on the travel times of the shared arcs because, unlike deterministic dominance, path travel time realizations are not used to compare the path travel time distribution functions.

Consider the example network in Figure 3.11 to illustrate the above argument.

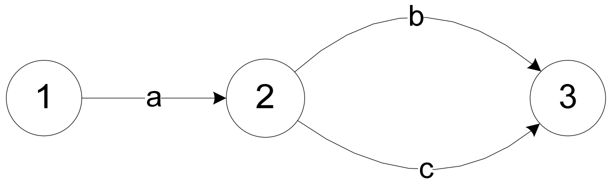


Figure 3.11. Example network

Table 3.4. Travel time pmf's for Figure 3.11

Arc a	**Arc b**		**Arc c**	
t=0	**t=1**	**t=2**	**t=1**	**t=2**
1(0.5)	3(0.4)	5(0.5)	3(0.38)	5(0.48)
2(0.5)	4(0.6)	6(0.5)	4(0.62)	6(0.52)

Arc *b* stochastically dominates arc *c* at both departure time 1 and 2 (the possible time at node 2). The paths' cdfs are determined and given in Table 3.5:

Table 3.5. Travel time cdf's of path ab and ac at departure time t=0 (cumulative probability)

Path ab	**Path ac**
t=0	**t=0**
4 (0.2)	4 (0.19)
5 (0.5)	5 (0.50)
7 (0.75)	7 (0.74)
8 (1.0)	8 (1.0)

Path *ab* dominates path *ac.* There is no need to condition on the travel times of shared arc *a* (at common departure time 0) because stochastic dominance is maintained when the path travel times cdf's are constructed using a shared arc.

Comparing two paths by their expected travel times is simply the comparison of two deterministic values. Let E[X$_1$(t)] be the expected time of the random variable for the travel time on Path 1, for departure time t; similarly for E[x$_2$(t)]. Then if E[X$_1$(t)] < E[x$_2$(t)], then Path 1 has a lower expected time than Path 2 at time interval t.

Deterministic dominance, stochastic dominance and comparison based on expected value are related as follows. By definition, deterministic dominance implies stochastic dominance, and stochastic dominance implies a lower expected value of the dominating path.

For a given departure time t,

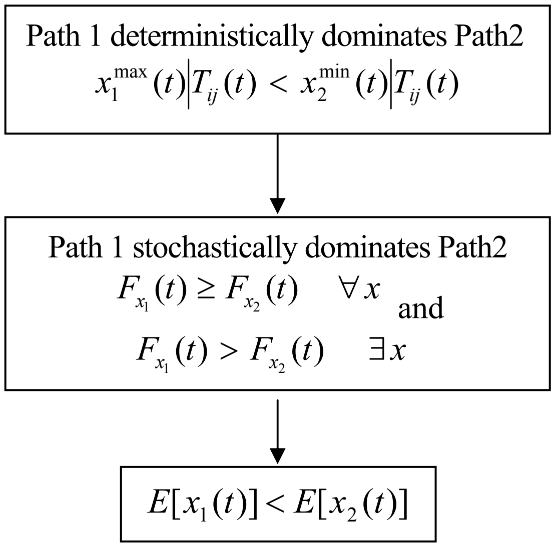

Path 1 deterministically dominates Path2
$x_1^{\max}(t)\big|T_{ij}(t) < x_2^{\min}(t)\big|T_{ij}(t)$
Path 1 stochastically dominates Path2
$F_{x_1}(t) \geq F_{x_2}(t) \quad \forall x$
and
$F_{x_1}(t) > F_{x_2}(t) \quad \exists x$
$E[x_1(t)] < E[x_2(t)]$

Figure 3.12. Three dominance criteria

3.3.2 Path Comparisons between Multiple Paths over A Time Period

When several paths exist between a pair of nodes, one or more of these paths may be dominated by at least one other path for every time interval in the period but by no single path for all time intervals. In any paired comparison of these paths, such a path will not be dominated. Dominance established by paired comparisons is referred to as pair-wise dominance. Pair-wise dominance applied to deterministic dominance, stochastic dominance and comparison via expected value is referred to as deterministic pair-wise dominance, stochastic pair-wise dominance, and expected value pair-wise dominance, respectively. For some applications, paths that are dominated by at least one other path for every time interval in the period are poorer paths (they will never be selected), even if they are non-dominated for every pairwise comparison. In this case, dominance can be determined by pairwise path comparisons at each time interval individually, referred to as group dominance. Here a dominated path is one that is dominated at each time interval in the period by at least one other path. By definition, the paths that are non-dominated by group dominance are all non-dominated by pairwise dominance. Deterministic group dominance, stochastic group dominance and expected value group dominance refer to dominance that is established by group comparisons for each time interval in the period by deterministic dominance, stochastic dominance and comparison via expected value, respectively. When multiple paths are compared over a time period, non-dominated optimal conditions based on group dominance will eliminate at least as many paths as conditions based on pairwise dominance.

An example is given next to illustrate the use of these dominance concepts for the comparison of three independent paths. The path travel times that have nonzero probability of occurring for each departure time in the peak period are shown in Table 3.6.

Table 3.6. Possible path travel times at time intervals 1 and 2

Path A		**Path B**		**Path C**	
Time 1	Time 2	Time 1	Time 2	Time 1	Time 2
4	5	6	8	3	4
5	9	7	7	6	6

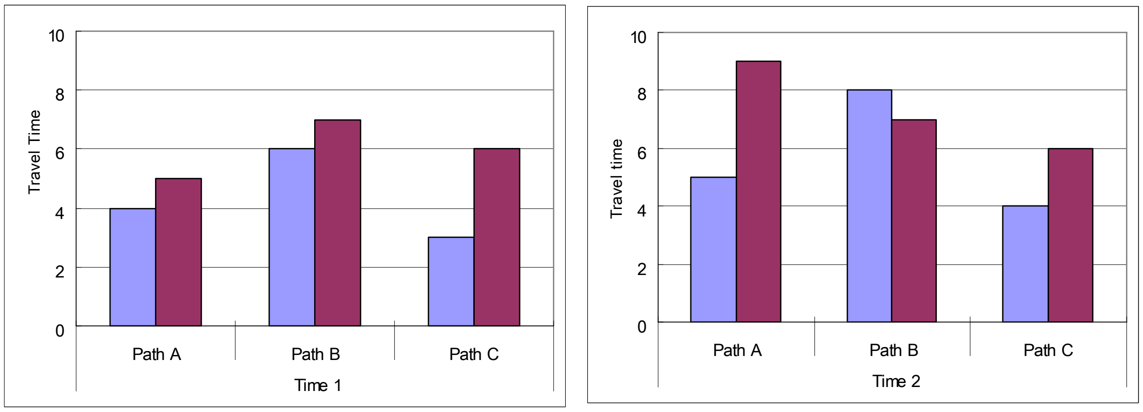


Figure 3.13. Path comparisons at time 1 and 2

It is seen from the Figure 3.13 that Path A deterministically dominates Path B at time interval 1 but that neither path deterministically dominates the other at time interval 2. Likewise, Path C deterministically dominates Path B at time interval 2 but neither path deterministically dominates the other at time interval 1. Thus, Path B is not dominated by Path A nor by Path C in both time intervals, but Path B is dominated in each time interval by one of either Path A or C. Considering all possible realizations of these paths for each time interval, it is seen in Tables 3.7.a and 3.7.b that Path B is never the least time path for any realization.

Table 3.7.a. Possible realizations for time interval 1

Realization	Path A	Path B	Path C	Best Path
1	4	6	3	C
2	4	6	6	A
3	4	7	3	C
4	4	7	6	A
5	5	6	3	C
6	5	6	6	A
7	5	7	3	C
8	5	7	6	A

Table 3.7.b. Possible realizations for time interval 2

Realization	Path A	Path B	Path C	Best Path
1	5	8	4	C
2	5	8	6	A
3	5	9	4	C
4	5	7	6	A
5	9	7	4	C
6	9	7	6	C
7	9	7	4	C
8	9	7	6	C

As illustrated in this example, a path that is not deterministically dominated by a single path over the entire time period, but is dominated at every time interval in the time period by at least one path, has zero probability of being the least time path for any realization of the network. Group dominance would eliminate those paths that have zero probability of being the least time paths for any realization. However, it is shown in Propositions 3.2 that group dominance is not sufficient for determining all non-dominated paths for a priori path selection, because some non-dominated paths may be incorrectly eliminated.

Proposition 3.2. Expected value group dominance is not sufficient for determining all least expected time paths in stochastic, time-dependent networks for a priori route selection.

Proof. (by counter example)

Assume that group dominance is sufficient to determine the least expected time path for a given departure time. For this given departure time, no dominated path can have a lower expected time than the nondominated path selected. A counterexample given in Figure 3.14 shows that it is possible that, for a given departure time, the least expected time path will be dominated if group dominance is permitted.

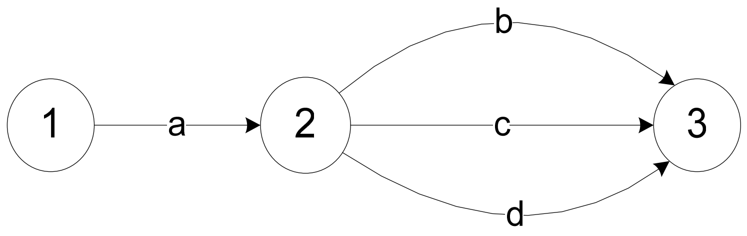


Figure 3.14. Example network

Table 3.8. Table of pmf s of travel times in Figure 3.14.

Arc *a*		**Arc *b***				**Arc *c***				**Arc *d***			
t=0		t=l		t=3		t=l		t=3		t=l		t=3	
1	0.5	4	0.6	5	0.4	4	1	9	1	8	0.7	3	1
3	0.5	6	0.4	7	0.6					12	0.3		

From Table 3.8, the expected travel times of all paths from node 1 to node 3 at departure time 0 are:

$E[ab]_{t=0}$=(1+4)*(0.5*0.6)+(1+6)*(0.5*0.4)+(3+5)*(0.5*0.4)+(3+7)*(0.5*0.6)=7.5 min

$E[ac]_{t=0}$ = (1+4)*0.5+(3+9)*0.5=8.5 min

$E[ad]_{t=0}$ = (1+8)*(0.5*0.7)+(1+12)*(0.5*0.3)+(3+3)*0.5=8.1 min.

The least expected time path from node 1 uses the path with constituent arcs *a* and *b*, Path *ab*. If group dominance is used then the path from node 2 to 3 with constituent arc *b*, Path *b*, would be eliminated because:

$E[b]_{t=1}$ = 4*0.6+6*0.4=4.8 minutes

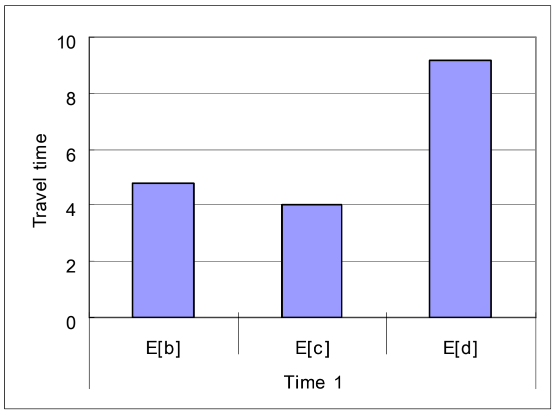
$E[b]_{t=3}$ = 5*0.4+7*0.6=6.2 minutes

$E[c]_{t=1}$ = 4.0 minutes

$E[c]_{t=3}$ = 9.0 minutes

$E[d]_{t=1}$ = 8*0.7+12*0.3=9.2 minutes

$E[d]_{t=3}$ = 3.0 minutes.



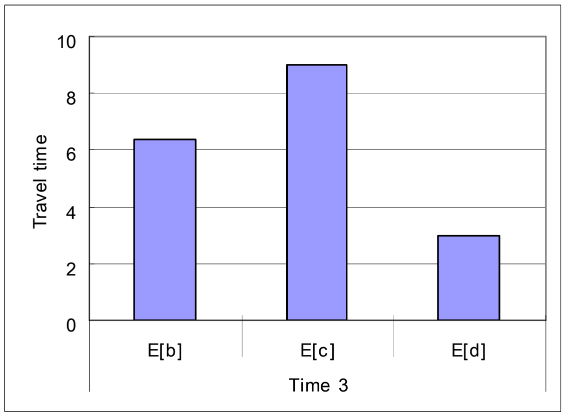


Figure 3.15. Arc travel time comparisons at time 1 and 3

Path *c* has a lower expected time than Path *b* at time interval 1 but not at time interval 2 and Path *d* has a lower expected time than Path *b* at time interval 3 but not at time interval 1. Thus, no path dominates Path *b* at both time intervals but Path *b* is dominated by some path at every time interval. Path *b* would be omitted from the set of solutions at node 2 and Path *ab* would be dominated, hence, omitting the least expected time path from the final solution. Therefore, expected value group dominance is not sufficient for determining least expected time paths.

In a priori path selection the entire route is selected before leaving the origin node. No deviations en route are permitted. Since the arc travel times are random variables, their actual values do not become known until travel along the arc has been completed. A better path can then be selected with this revealed information, referred to by Hall (1986) as "time-adaptive route choice." A similar concept arises in solving the transit equilibrium assignment problem where transit users have a set of strategies for determining which route to take in light of intermediate information, such as the bus arrival times at transfer points (Nguyen and Pallottino, 1988; Spiess and Florian, 1989; Wu and Florian, 1993; Wu et al., 1994). Such strategies can be represented by an acyclic subnetwork, called a hyperpath, that includes the arcs corresponding to these strategies with their associated conditional probabilities of being selected (Nguyen and Pallottino, 1986, 1988). In the context of time-adaptive route selection in stochastic, time-dependent networks, for each departure time interval, the set of optimal (for least expected time) strategies for selecting the best path between a pair of nodes can be depicted as a hyperpath where the conditional probabilities associated with the arcs may be replaced by conditional expected travel times or conditional travel time probability distribution functions for each departure time interval.

Using the example in the proof of Proposition 3.2, it is shown that for a given departure time interval, the least expected time paths for a priori path selection are not necessarily the paths with the least expected time in a time-adaptive route choice framework. Consider the expected time on each Path *ab*, *ac* and *ad*, at departure time 0 from node 1. For departure time 0, the path with the least expected time of 7.5 minutes is *ab*. However, for any possible travel time on arc *a*, and thus, any possible arrival time at node 2, arc *b* is never on the least expected time path:

E[*ab*/*a*=1] = 5.8 minutes and E[*ab*/*a*=3] = 9.2 minutes

E[*ac*/*a*=1] = 5 minutes and E[*ac*/*a*=3] = 12 minutes

E[*ad*/*a*=1] = 10.2 minutes and E[*ad*/*a*=3] = 6.0 minutes.

For the example problem, for departure time 0 from node 1, the least expected time path is Path *ac* if the driver arrives at node 2 in departure time interval 1 and it is Path *ad* if the driver arrives there in departure time interval 2. Path *ab*, the a priori least expected time path for departure time 0, is not the least expected time path when the driver is permitted to react to the actual (revealed) arrival time at node 2. By group dominance the a priori least expected time paths that contain subpaths to the destination node that are never the least expected time subpaths for any possible arrival times are eliminated. For each departure time interval, the set of optimal strategies with the associated conditional expected values or pdf's, respectively, be identified from this reduced set of paths.

3.4 Conclusions

In this chapter, concepts and path selection criteria are described for networks where arc times are random variables with time-dependent probability distribution functions. In the next chapter, procedures based on the concepts of expected value dominance described in this chapter are presented for determining non-dominated least expected time paths in stochastic, time-dependent networks.

Chapter 4. A Priori Minimum Variance and Mean-Variance Path in Stochastic Time-Dependent Networks

In this chapter, the algorithm for determining a minimum variance travel time path in a stochastic time dependent networks for a priori path selection are presented. In section 4.1, we present additional notation for stochastic time dependent networks. The notation of this subsection complements the notation given in subsection 3.1. A mathematical formulation for the problem of computing minimum expected travel time and variance travel time from all nodes to a given destination node *d* were described in section 4.2 and 4.3. In section 4.4 and 4.5, procedures, referred to as the PMV and PMMV algorithms, are presented for determining a priori minimum variance path and minimum mean-variance path from all nodes to a given destination node for all time intervals. Concluding remarks are given in Section 4.6.

4.1 Additional Notation for Stochastic Time-Dependent Networks

Assume that link travel times are discrete random variables with time-dependent mass probability functions. We assume independence across the arcs and time of all random variables. $T_{ij}(t)$ denotes the discrete random variables corresponding to the travel time of link (i, j) at time t. The function $\tau_{ij}^{k}(t),\ p_{ij}^{k}(t)$ denotes the probability mass function corresponding to the random variable $T_{ij}(t)$, where $k_{ij}(t)$ denotes the number of distinct possible values for $T_{ij}(t)$ and $k \in \{1, 2, ..., k_{ij}(t)\}$. The probability that

$T_{ij}(t)$ takes the value $\tau_{ij}^{k}(t)$ is $p_{ij}^{k}(t)$. The maximum range of $k_{ij}(t)$, over (*i, j, t*) is denoted by R. We assume that all possible link travel times are strictly positive and finite.

In order to ensure the representation of all time-dependent network data within a finite computer memory, we impose the common restriction that all such data is only specified within the finite time window $t \in \{0, 1, 2, ..., M\}$. This window must be made large enough to capture any relevant time-dependency in link travel time data in a real-world application. For times after the time horizon, M, all link travel times are assumed to be static and equal to the value they are assumed at time *M*; that is, $\tau_{ij}^{k}(t \geq M) = \tau_{ij}^{k}(M)$ and $p_{ij}^{k}(t \geq M) = p_{ij}^{k}(M)$.

In the algorithm of this section, since link travel times are assumed to be positive, negative-cost cycles may not arise in the dynamic region of time, corresponding to $t<M$. They may however arise in the static region of time, corresponding to time instant greater than or equal to *M*. We then impose the common restriction that there be no negative-cost cycles in the network, for $t \geq M$. Throughout this study we will treat the entire set of times $\{t: t \geq M\}$ as an atomic unit. We have assumed that the set of discrete times and link travel times are integers. The actual discrete time set however, need not to be the set of integers. The adopted discrete-time assumption is common in literature, and includes the representation of actual time instants and travel times that are multiples of the value of this constant spacing. The value of parameter *M* can then be viewed as the number of time sub-intervals resulting from the discretization of a given time period, such as the peak-period in a traffic network, using a given time spacing. A finer time-discretization would lead to increased accuracy in a given network model and an increase in the value

of parameter *M*.

4.2 Problem Formulation

4.2.1 Expected Travel Time A Routing Problem

We are interested in developing a mathematical formulation for the problem of computing least expected travel times, and a corresponding solution, from all nodes to a given destination node *d*. Let a random variable $L_i(t)$ denote the travel time from node *i* to destination node *d*, considering that one departs from node *i* at time *t*. Similarly, let random variable $L_{ij}(t)$ denote the travel time to destination node d, if one departs at the beginning of arc (i, j) at time *t*. The expected values of $L_i(t)$ and $L_{ij}(t)$ are respectively denoted $e_i(t)$ and $e_{ij}(t)$ where $e_i(t) = E[L_i(t)]$ and $e_{ij}(t) = E[L_{ij}(t)]$.

For all t >= 0, we have the following relation:

$$e_{ij}(t) = E[L_{ij}(t)] = E[T_{ij}(t)] + E[L_j(t + T_{ij}(t))] \qquad (4\text{-}1)$$

The expected value of $L_j(t + T_{ij}(t))$ is given by:

$$\begin{aligned} E\left[L_j\left(t + T_{ij}(t)\right)\right] &= \sum_{k}^{k_{ij}(t)} E\left[L_i\left(t + \tau_{ij}^k(t)\right)\right] * p_{ij}^k(t) \\ &= \sum_{k}^{k_{ij}(t)} e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t) \end{aligned} \qquad (4\text{-}2)$$

Hence, the minimum expected travel time $e_{ij}(t)$ is given by:

$$\begin{aligned} e_{ij}(t) = E(L_{ij}(t)) &= E\left[T_{ij}(t)\right] + E\left[L_j(t + T_{ij}(t))\right] \\ &= \sum_{k}^{k_{ij}(t)} \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_{k}^{k_{ij}(t)} \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right] \qquad (4\text{-}3) \\ &= E\left[\tau_{ij}^k(t) + e_j(t + \tau_{ij}^k(t))\right] \end{aligned}$$

Note that if *t*>=M, expression (4-3) changes to:

$$\begin{aligned} e_{ij}(t) &= \sum_{k}^{k_{ij}(t)} \left[\tau_{ij}^k(M) * p_{ij}^k(M)\right] + \sum_{k}^{k_{ij}(t)} \left[e_j\left(M\right) * p_{ij}^k(M)\right] \\ &= E\left[T_{ij}(M)\right] + e_j(M) \qquad (4\text{-}4) \end{aligned}$$

The minimum expected travel times $e_i(t)$ then verify the following functional equations:

$$e_i(t \geq M) = e_i(M) = \begin{cases} \min\limits_{j \in A(i)} \left(E\left[T_{ij}(M)\right] + e_j(M)\right) & i \neq d \\ 0 & i = d \end{cases} \qquad (4\text{-}5)$$

$$e_i(t) = \begin{cases} \min\limits_{j \in A(i)} \left(\sum_{k}^{k_{ij}(t)} \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_{k}^{k_{ij}(t)} \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right] \right) & i \neq d,\ \forall t < M \\ 0 & i = d,\ \forall t < M \end{cases} \qquad (4\text{-}6)$$

Now denote by $\pi_i(t)$ a node such that (*i,* $\pi_i(t)$) is a next "best" arc corresponding to minimum expected travel time $e_i(t)$. Functions $\pi_i(t)$ verify the following equations:

$$\pi_i(t \geq M) = \pi_i(M) = \begin{cases} Arg \min\limits_{j \in A(i)} \left(E\left[T_{ij}(M)\right] + e_j(M)\right) & i \neq d \\ 0 & i = d \end{cases} \qquad (4\text{-}7)$$

$$N_i(t) = \begin{cases} Arg \min\limits_{j \in A(i)} \left(\sum_{k}^{k_{ij}(t)} \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_{k}^{k_{ij}(t)} \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right] \right) & i \neq d,\ \forall t < M \\ 0 & i = d,\ \forall t < M \end{cases} \qquad (4\text{-}8)$$

Functional equations (4-5)-(4-8) define a formulation to the problem of computing the all-to-one minimum expected travel times and an associated both next arcs solution corresponding to the "best" next arc routing policy and a priori best path routing policy.

Remark:

Deterministic dynamic networks can be viewed as a particular case of stochastic dependent networks, where for all (i, j, t) we have: $k_{ij}(t) = 1, \left(\tau_{ij}^{k}(t),\ p_{ij}^{k}(t)\right) = \left(\tau_{ij}(t),\ 1\right)$. Hence, functional equations (4-5)-(4-6) are equivalent to:

$$e_i(t \geq M) = e_i(M) = \begin{cases} \min\limits_{j \in A(i)} \left(T_{ij}(M) + e_j(M)\right) & i \neq d \\ 0 & i = d \end{cases} \tag{4-9}$$

$$e_i(t) = \begin{cases} \min\limits_{j \in A(i)} \left(\tau_{ij}(t) + e_j\left(t + \tau_{ij}(t)\right)\right) & i \neq d,\ \forall t < M \\ 0 & i = d,\ \forall t < M \end{cases} \tag{4-10}$$

Functional equations (4-9)–(4-10) are necessary and sufficient optimality conditions of the all-to-one minimum travel-time path problem in deterministic dynamic networks (see for instance, Cooke and Halsey (1969) or Chabini (1998) for more details about proving this equivalency and developing efficient solution algorithms). We indicated that in deterministic time dependent networks, the "best" path routing policy and the "best" next arc routing policy are equivalent.

4.2.2 The Optimality Condition for Mean

Define N(i) as the set of downstream nodes of node i, $\tau_{ij}^{k}(t)$ as the travel time random variable for link (i, j) at time *t*. We make the assumption that there exists at least one path from any node to the destination node *d* under any possible value of the link travel time vector. $e_i(t)$ and $\pi_i(t)$ are optimal if and only if they are solutions of the following system of equations:

$$e_i(t) = \min_{j \in A(i)} \left\{ E\left[\tau_{ij}(t) + e_j(t + \tau_{ij}(t)) \right] \right\}$$

$$\pi_i(t) = \arg\min_{j \in A(i)} \left\{ E\left[\tau_{ij}(t) + e_j(t + \tau_{ij}(t)) \right] \right\}$$

with the boundary conditions:

$$e_d(t) = 0,\ \pi_d(t) = d,\ \forall t \in T \quad \text{and} \quad e_i(t) = e_i(M),\ \forall i \in N,\ t > M$$

Note that we assume the outcome of the decision is deterministic, i.e. the traveler will end up at node j if he/she chooses node j as his/her next node. Croucher (1978) studies the problem where the outcome of the decision itself is stochastic. We do not discuss this case, as our motivation in studying the optimal routing problem is for traffic applications, where this case rarely arises.

The proof of the optimality conditions is similar to the proof of Proposition 7.2.1 in Bertsekas (2000). The problem in Bertsekas (2000) is denoted as a stochastic shortest path problem and is viewed as an infinite horizon dynamic programming problem. The proof provided uses only the node number as a state, yet we can simply replace the state by {*i, t*} and the proof becomes valid for our case.

We will show an illustrative example of how the optimality condition works. The topological network is shown at the upper side of Figure 4.1, and the major part of the figure represents a time-space representation of the network. In a time-space network, time is shown along the vertical axis (the time axis), and the node number is shown along the horizontal axis (the space axis).Each point in this network represents a node-time pair *(i, t)*, and any link between *(i, t_1)* and *(j, t_2)* indicates that link *(i, j)* has a travel time of $t_2 - t_1$ if departure time from node *i* is t_1.We are interested in finding the minimum expected travel time path from node 1 to node 4 at departure time 0, namely $e_1(t = 0)$, and only these node-time pairs and links which are relevant to the computation are shown.

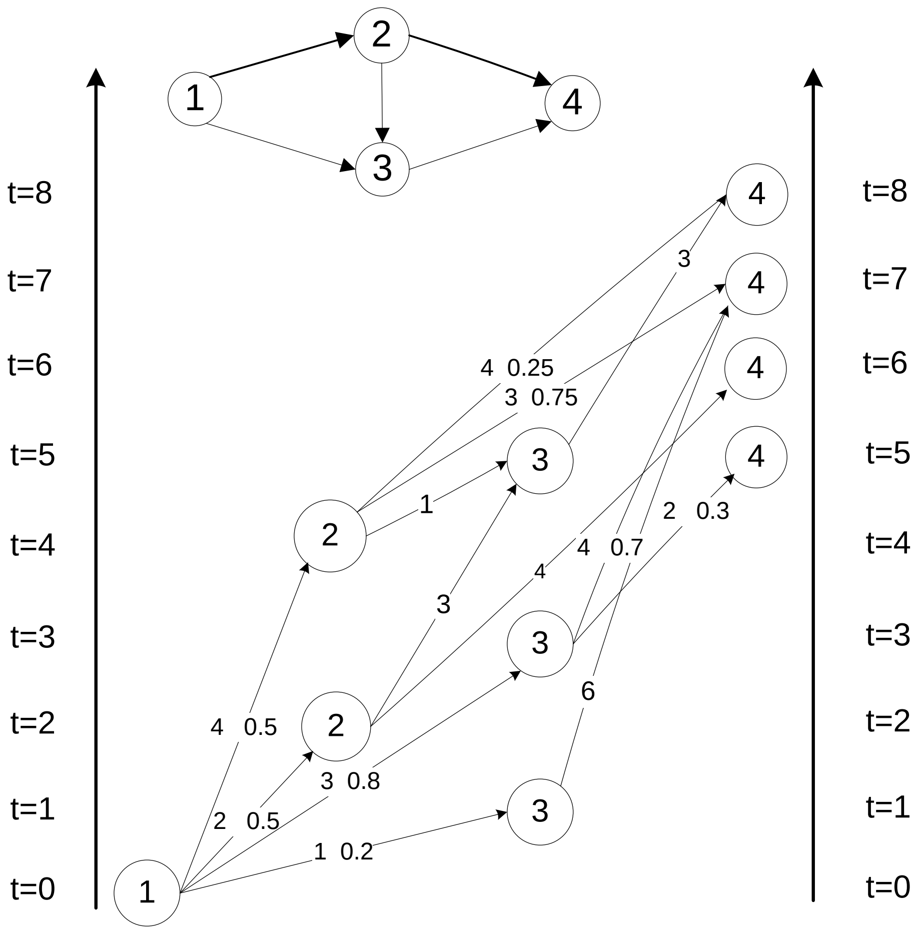


Figure 4.1. An Illustrative Example for Optimality Conditions

Figure 4-1 shows the marginal distributions of the link travel time random variables. Link (1, 2) at time 0 could have two values of travel time: 4 with probability 0.5 and 2 with probability 0.5. Link (1, 3) at time 0 could have two values of travel time: 1 with the probability 0.2 and 3 with the probability 0.8. Link (2, 4) at time 4 could have two values of travel times: 3 with probability 0.75 and 4 with probability 0.25. Link (3, 4) at time 3 could have two values of travel times: 2 with probability 0.3 and 3 with probability 0.7. All other link travel times are deterministic.

We apply the optimality conditions to obtain the value of $e_1(t=0)$.

$$e_1(t=0)=\min\{(0.2*(1+e_3(1)))+(0.8*(3+e_3(3))),\ (0.5*(2+e_2(2)))+(0.5*(4+e_2(4)))\}$$

It can be easily observed from the figure that $e_3(t=1)$ =6 and $\pi_3(1)$ =(node)4, $e_2(t=2)$ =4 and $\pi_3(1)$ =(node) 4, $e_3(t=5)$ = 3 and $\pi_3(5)$ =(node)4, and $e_3(t=3)$=2*0.3+4*0.7=3.4 and $\pi_3(3)=$(node)4. We apply the optimality condition again to obtain $e_2(t=4)$:

$$e_2(t=4)=\min\{1+e_3(5),\ 0.25*4+0.75*3\}=\min\{6,\ 3.25\}=3.25$$

and $\pi_2(4)$=(node)4. With the values of $e_3(t=1)$, $e_2(t=2)$, $e_3(t=3)$, and $e_2(t=4)$ in hand, we can obtain

$$e_1(t=0)=\min\{(0.2*6)+(0.8*(2+3.4)),\ (0.5*(2+4))+(0.5*(4+3.25))\}\text{=5.52}$$

and $\pi_1(0)=3$. Therefore, minimum expected travel time for node 1 at time 0 turn out to be a path: 1-3-4.

4.2.3 Variance of a Routing Problem

Before presenting the optimality conditions, we try to find the recursive relationship between the variances of a given routing policy starting from two adjacent nodes. This relationship is much more involved than that for the expected travel time of a routing policy. As we know, the expected travel time of a routing can be decomposed into two parts: one is the expected travel time of the next link, and the other is the expected travel time from the next state (whose current-node is the next node) to the destination.

As we have seen in previous section, $e_i(t)$ denotes the expected travel time from i to the given destination node *d*. We define $T_{ij}(t)$ as a travel time random variable of link (i, j) at time t conditional on current travel time information, and $L_j(t+T_{ij}(t))$ as a travel time from node *j* to destination node *d* at time $t+T_{ij}(t)$, if one departs from node *i*. Then we have

$$e_i(t) = E[T_{ij}(t) + E\left(e_j(t+T_{ij}(t))\right)]$$

Next, we develop the recursive equation for the variance of a routing. We define additional variables as follows. All routing decisions are made to reach a single destination *d*.

$v_i(t)$: travel time variance from node *i* to destination node *d* at time *t*

$v_i(t) = Var[L_i(t)]$

$p_{ij}(t)$: probability that takes the value $\tau_{ij}(t)$

In the following mathematical development, all the calculations are conditional on the current travel time information. The major theorem we use is the Law of Conditional Variances(Ross, 1989):

By definition of Var(X/Y), we have that

$$\begin{aligned} Var(X/Y) &= E[(X - E(X/Y))^2] \\ &= E[X^2 - 2XE(X/Y) + E^2(X/Y)\ /Y] \qquad (4\text{-}11) \\ &= E[X^2/Y] - 2E[X/Y]E[X/Y] + E^2[X/Y] \end{aligned}$$

where we the fact that E[X/Y] and E^2[X/Y] are functions of Y and thus, given Y, they may be treated as constants.

Therefore,

$$\begin{aligned} Var(X/Y) &= E[X^2/Y] - 2E[X/Y]E[X/Y] + E^2[X/Y] \\ &= E[X^2/Y] - E^2[X/Y] \end{aligned} \tag{4-12}$$

and taking expectations yields

$$\begin{aligned} E[Var(X/Y)] &= E[\,E[X^2/Y]] - E[E^2[X/Y]] \\ &= E[X^2] - E[E^2[X/Y]] \end{aligned} \tag{4-13}$$

$$\begin{aligned} Var(E[X/Y]) &= E[(E[X/Y] - E(E[X/Y]))^2] \\ &= E[(E(X/Y) - E(X))^2] \\ &= E[E^2[X/Y]] - 2E[E[X/Y]]E[X] + E^2[X] \\ &= E[E^2[X/Y]] - 2E^2[X] + E^2[X] \\ &= E[E^2[X/Y]] - E^2[X] \end{aligned} \tag{4-14}$$

Hence, from equation (4-13) and (4-14), we arrive at

$$\begin{aligned} E[Var(X/Y)] + Var(E[X/Y]) &= E[X^2] - E[E^2[X/Y]] + E[E^2[X/Y]] - E^2[X] \\ &= \ E[X^2] - E^2[X] \\ &= Var(X) \end{aligned} \tag{4-15}$$

Therefore, the variance of X, given the random variable Y, is defined by

$$Var(X) = E[Var(X/Y)] + Var(E[X/Y])$$

Note that $E[X/Y]$ and $Var(X/Y)$ are also random variables. $E[X/y]$ is a constant, which is the expected value of X given that Y = y. $Var(X/y)$ is a constant, which is the variance of X given that Y = y.

Since $L_i(t) = T_{ij}(t) + L_j(t + T_{ij}(t))$, we have

$$\begin{aligned} v_i(t) &= Var[L_i(t)] \\ &= E[Var[L_i(t)/T_{ij}(t)] + Var[E[L_i(t)/T_{ij}(t)]] \\ &= \sum_k p_{ij}^k(t) * Var[L_i(t)/\tau_{ij}] + Var[E[L_i(t)/T_{ij}(t)]] \end{aligned} \tag{4-16}$$

The first equality is according to the definition of $v_i(t)$. The second equality is due to the Law of Conditional Variances. The third equality is according to the definitions of expected value and the variance of a random variable.

Next, we compute the individual components of the right hand side of the last line in Equation 4.3 one by one.

We apply the Law of Conditional Variances again to obtain

$$\begin{aligned} Var[L_i(t)/\tau_{ij}(t)] &= Var[T_{ij}(t) + L_j(t+\tau_{ij}(t))] \\ &= Var[L_j(t+\tau_{ij}(t)] \\ &= v_j(t+\tau_{ij}(t)) \end{aligned} \tag{4-17}$$

The first equality is due to the decomposition of travel time from (i, t) into two parts. The second equality is due to the fact that $\tau_{ij}(t)$ is a deterministic value and thus, does not contribute to the variance of $L_i(t)$. The third equality is by the definition of $v_i(t) = Var[L_i(t)]$.

Therefore we have

$$\begin{aligned} E[Var[L_i(t)/\tau_{ij}(t)] &= \sum_k p_{ij}^k(t) * Var[L_i(t)/\tau_{ij}(t)] \\ &= \sum_k p_{ij}^k(t) * v_j(t+\tau_{ij}(t)) \end{aligned} \tag{4-18}$$

Now that we have finished developing the first component of the right hand side of the last line of Equation (4-16), let us study the second component.

$$\begin{aligned} E[L_i(t)/\tau_{ij}(t)] &= \tau_{ij}(t) + E[L_j(t+\tau_{ij}(t))] \\ &= \tau_{ij}(t) + e_j(t+\tau_{ij}(t)) \end{aligned} \tag{4-19}$$

Therefore the expectation of $E[L_i(t)/\tau_{ij}(t)]$ is evaluated as:

$$\begin{aligned} E[E[L_i(t)/\tau_{ij}(t)]] &= \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t+\tau_{ij}(t))\right) \\ &= e_i(t) \end{aligned} \tag{4-20}$$

and the second component of the right hand side of the last line of Equation (4-16), which is actually the variance of $E[L_i(t)/\tau_{ij}(t)]$, can be evaluated as:

$$\begin{aligned} Var[E[L_i(t)/T_{ij}(t)]] &= E[E[L_i(t)/T_{ij}(t)] - E[E[L_i(t)/T_{ij}(t)]]]^2 \\ &= \sum_k p_{ij}^k(t) * \left(E[L_i(t)/T_{ij}(t)] - E[E[L_i(t)/T_{ij}(t)]]\right)^2 \qquad (4\text{-}21) \\ &= \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t+\tau_{ij}(t)) - e_i(t)\right)^2 \end{aligned}$$

Substituting Equation (4-18) and (4-21) into Equation (4-16), we obtain the final result:

$$\begin{aligned} v_i(t) &= Var[L_i(t)] \\ &= E[Var[L_i(t)/T_{ij}(t)] + Var[E[L_i(t)/T_{ij}(t)]] \qquad (4\text{-}22) \\ &= \sum_k p_{ij}^k(t) * v_j(t+\tau_{ij}(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t+\tau_{ij}(t)) - e_i(t)\right)^2 \end{aligned}$$

$$v_d(t) = 0$$

Please note that all calculations are conditional on the current travel time information. Intuitively, we can view the first part as the variance from the next node to the destination, and the second part as the variance induced by including the next link in the routing.

For the given origin node i and departure time t , the objective of the “minimum variance path” problem is to minimize the variance of travel time to a select destination given that the path can be determined before travel starts. Let $v_i(t)$ be the minimum travel time variance from node i to destination node d at departure time t. The problem is then to find the best path routing , that is, the set of paths, for each origin node at each departure time in the peak period such that $v_i(t)$ is minimized. As we discussed in previous section 4.2, $v_i(t)$ can be computed as follows:

$$v_i(t) = \begin{cases} \min\limits_{j\in A(i)} \left(\sum_k p_{ij}^k(t) * v_j(t+\tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t+\tau_{ij}^k(t)) - e_i(t)\right)^2 \right) & ; i \neq d, \ \forall t < M \\ 0 & ; i = d, \ \forall t < M \end{cases} \qquad (4\text{-}23)$$

$$v_i(t \geq M) = \begin{cases} v_i(t=M) & ; i \neq d, \forall t \geq M \\ 0 & ; i = d, \forall t \geq M \end{cases}$$

Now denote by $\pi_i(t)$ a node such that (i, $\pi_i(t)$) is a next arc corresponding to minimum travel time variance $v_i(t)$. Functions $\pi_i(t)$ verify the following equation:

$$\pi_i(t) = \begin{cases} Arg\min_{j \in A(i)} \left(\sum_k p_{ij}^k(t) * v_j(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t + \tau_{ij}^k(t)) - e_i(t)\right)^2 \right) & ; i \neq d, \forall t < M \\ 0 & ; i = d, \forall t < M \end{cases} \quad (4\text{-}24)$$

$$\pi_i(t) = \begin{cases} \pi_i(t=M) & ; i \neq d, \forall t \geq M \\ 0 & ; i = d, \forall t \geq M \end{cases}$$

Functional equations (4-23) and (4-24) define a formulation to the problem of computing the all-to-one minimum travel time variance.

To the best of the author's knowledge, there is no paper in the literature that deals with minimum variance path (routing) problems in stochastic time dependent networks. The study here is a preliminary attempt to tackle the minimum mean-variance routing problem.

4.2.4 The optimality Condition for variance

We make the assumption that there exists at least one path from any node to the destination node *d* under any possible value of the link travel time variance. $v_i(t)$ and $\pi_i(t)$ are optimal if and only if they are solutions of the following system of equations:

$$v_i(t) = \min_{j \in A(i)} \left(\sum_k p_{ij}^k(t) * v_j(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t + \tau_{ij}^k(t)) - e_i(t)\right)^2 \right)$$

$$\pi_i(t) = \arg\min_{j \in A(i)} \left\{ \left(\sum_k p_{ij}^k(t) * v_j(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t + \tau_{ij}^k(t)) - e_i(t)\right)^2 \right) \right\}$$

with the boundary conditions:

$$\nu_d(t) = 0,\ \pi_d(t) = d,\ \forall t \in T \quad \text{and} \quad \nu_i(t) = \nu_i(M),\ \forall i \in N,\ t > M$$

Note that we assume the outcome of the decision is deterministic, i.e. the traveler will end up at node *j* if he/she chooses node j as his/her next node.

We will show an illustrative example of how the optimality condition works.

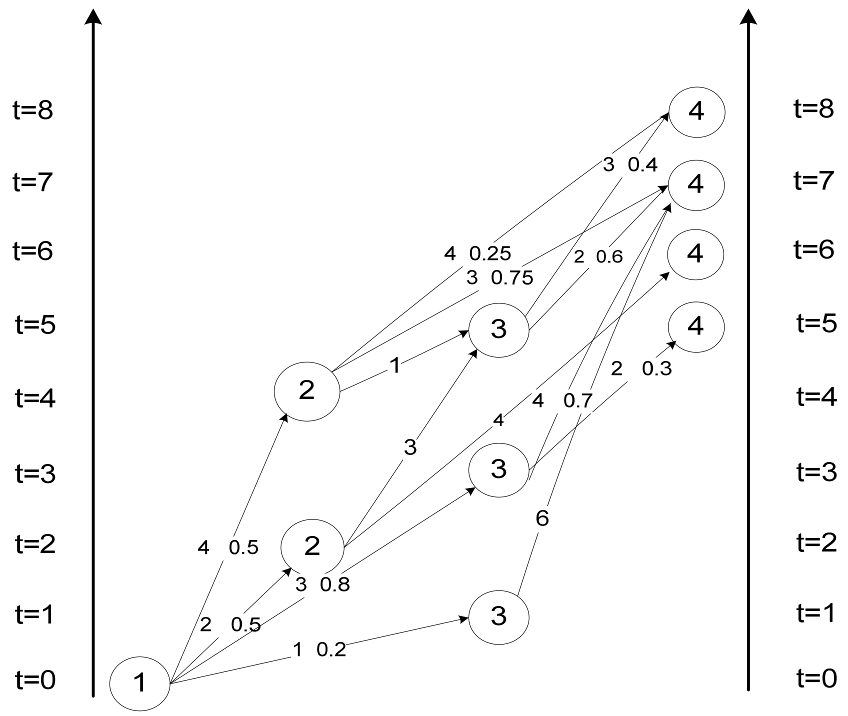


Figure 4.2. An Illustrative Example for Optimality Conditions

The topological network is shown at the upper side of Figure 4-1, and the major part of the Figure is a time-space representation of the network. In a time-space network, time is shown along the vertical axis (the time axis), and the node number is shown along the horizontal axis (the space axis).Each point in this network represents a node-time pair (i, t), and any link between (i, t_1) and (j, t_2)indicates that link (i, j) has a travel time of $t_2 - t_1$ if departure time from node *i* is t_1.We are interested in finding the minimum variance path from node 1 to node 4 at departure time 0, namely $v_1(t=0)$, and only those node-time pairs and links relevant to the computation are shown.

Figure 4.2. shows the marginal distributions of the link travel time random variables. Link (1, 2) at time 0 could have two values of travel time: 4 with probability 0.5 and 2 with probability. 0.5. Link (1, 3) at time 0 could have two values of travel time: 1 with probability 0.2 and 3 with probability 0.8. Link (2, 4) at time 4 could have two values of travel times: 3 with probability 0.75 and 4 with probability 0.25. Link (3, 4) at time 3 could have two values of travel times: 2 with probability 0.3 and 3 with probability 0.7. All other link travel times are deterministic.

We apply the optimality conditions to obtain the value of $e_1(t=0)$.

$$e_1(t=0)=\min\left\{\begin{array}{l}\begin{pmatrix}(0.5*v_2^1(2))+(0.5*(2+e_2^1(2)-e_1^1(0))^2\\+(0.5*v_2^1(4))+(0.5*(4+e_2^1(4)-e_1^1(0))^2\end{pmatrix}_{path1-2-4},\\ \qquad\begin{pmatrix}(0.5*v_2^2(2)+(0.5*(2+e_2^2(2)-e_1^2(0))^2\\+(0.5)*v_2^2(4)+(0.5)*(4+e_2^2(4)-e_1^2(0))^2\end{pmatrix}_{path1-2-3-4}\\ \qquad\qquad\begin{pmatrix}(0.2)*v_3^1(1)+(1*0.2)*(1+e_3^1(1)-e_1^3(0))^2\\+(0.8)*v_3^1(3)+(3*0.8)*(4+e_3^1(3)-e_1^3(0))^2\end{pmatrix}_{path1-3-4}\end{array}\right\}$$

It can be easily observed from the Figure that

$$e_2^1(t=2)=4,\ \ v_2^1(t=2)=0$$

$$e_2^1(t=4)=3.25,\ \ v_2^1(t=4)=0.1875$$

$$e_1^1(t=0)=6.625$$

$$e_2^2(t=2)=5.4,\ \ v_2^2(t=2)=0.24$$

$$e_2^2(t=4)=3.4,\ \ v_2^2(t=4)=0.24$$

$$e_1^2(t=0)=7.4$$

$$e_3^1(t=1)=6,\ \ v_3^1(t=1)=0$$

$$e_3^1(t=3)=3.4,\ \ v_3^1(t=3)=0.84$$

$$e_1^3(t=0) = 6.52$$

Therefore, $v_1(t=0) = \min\left\{(0.4844)_{path\ 1-2-4},\ (2.24)_{path\ 1-2-3-4},\ (0.7296)_{path\ 1-3-4}\right\} = 0.4844$ and $\pi_1(0) = 2$, p=1. Therefore, minimum variance of travel time for node 1 at time 0 turn out to be a path: 1-2-4.

4.3 An Algorithm for A Priori Minimum Variance Path Problems

In this section, an algorithm to compute minimum variance path routing policies with a criterion of travel time reliability (variance) were presented.. We have been focusing on the study of minimum expected travel time policies, as expected travel time is the primary concern of travelers in making routing decisions. On the other hand, when faced with uncertainty, travelers are also concerned about the reliability of their travel times. For example, unreliable travel times will cause anxiety or disutility among travelers because of the possibility of an unexpected late arrival at their destinations. We use travel time variance to represent travel time reliability. A routing policy with less travel time variance is viewed as more reliable. For commuters, the desired arrival time in the morning might be some time around the work starting time. For a traveler catching a plane, the desired arrival time might be roughly one hour before the plane's departure. It is generally believed and verified by some empirical studies that both early and late arrivals cause disutility to the user. For example, although late arrival at the workplace would cause trouble for a commuter, an arrival too early would also make the commuter feel as if it was a waste of time.

Therefore, we design algorithms that minimize travel time variance from all nodes to a given destination node *d*. We develop formulas that describe the relationship between a variance at a given state *(i, t)* and the attributes at succeeding nodes. Then we present the optimal condition for the policy that minimizes the travel time variance. The following two sections provide a theoretical base for the algorithm design in next section. The illustrative examples are presented to have good picture of two algorithms.

4.3.1 An Algorithm

For each node i∈N and each potentially optimal path *h* to the destination node *d*, a vector label $\xi_i^h(t), t \in T$ is maintained, where $\xi_i^h(t), t \in T$ is the expected travel time along path *h* from node *i* to the destination, leaving node *i* at time *t*; i.e., $\xi_i^h(t) = E[L_i^h(t)]$. Similarly, a vector label $\omega_i^h(t), t \in T$ is maintained for the variance of travel time along path *h* from node *i* to the destination, leaving node *i* at time *t*; i.e., $\omega_i^h(t) = Var[L_i^h(t)]$.

These labels are called candidate-optimal because each is potentially optimal for one or more time intervals. Until the termination of the algorithm, more than one label vector is maintained at each node unless a single label is best for all time intervals. Let *q*(i) be the set of candidate-optimal labels at node i. At each iteration of the algorithm, a node *j* is scanned and a temporary label vector is constructed, $\xi_i^h(t) \leftarrow e_i^h(t)$ for expected time and $\omega_i^h(t) \leftarrow v_i^h(h)$ for variance, from each of its predecessor nodes, i∈A(i, j). This temporary label is compared with the candidate-optimal labels at node i, Ω_j(t), according to the following conditions:

$\omega_i^h(t)$ corresponds to a candidate-optimal path iff $\exists$ no path $h \in q(i)$

such that $\Omega_i(t) \le \omega_i^h(t) \;\; \forall t \in T$,otherwise the path is dominated.

Even if a temporary label is dominated by any currently candidate-optimal path, this temporary label can be a part of another candidate-optimal path. Therefore, all temporary labels need to be kept for future path variance calculations. This is major difference between a priori least expected time path algorithm and minimum variance path algorithm.

Two pointers are required for each label c at each node i to store the candidate-optimal paths efficiently: a pointer, π_i^h (t), from the hth label at node *i* to the next node on the path and a pointer, $\theta_i^h(t)$, to indicate the appropriate path label at the next node. Note that π_i^{temp} and θ_i^{temp} hold the path information of a temporary label until that label is determined to be non-dominated solution or is discarded.

Algorithm PMV

```
begin
    Procedure Initialization
        begin
            create the NODE_LIST, SA
            put all nodes i to NODE_LIST
            set SA=Ø
            set each node i,
                $\xi_i^h(t) = \infty, \; \forall i \in N-d, \, t \in T, \, h \in \{1,2,...,P\}$
                    where P is a large enough number to permit as many
                    candidate-optimal path at any node as might be required
                $\xi_d^h(t) = 0, \; \forall t \in T$
                $\omega_i^h(t) = \infty, \; \forall i \in N-d, \, t \in T, \; h \in \{1,2,...,P\}$
                $\omega_d^h(t) = 0, \; \forall t \in T$
            $\Omega_i(t) = \infty,\ \forall i \in N - d,\ t \in T,\ h \in \{1,2,...,P\}$
            $\Omega_d(t) = 0,\ \forall t \in T$
            $\pi_i^h(t) = \infty,\ \forall i \in N, t \in T,\ h \in \{1,2,...,P\}$
            $\theta_i^h(t) = \infty,\ \forall t \in T\ ,\ h \in \{1,2,...,P\}$
            $q$(d) = 1 (put the first path label at node d)
        Insert destination node and path label pair (*d, 1*) to set SA list
      **end**
**while** SA≠Ø do
  **begin**
      select the first node and path label from the set SA
      call this node the current node, *j*
      scan the current node, *j*
        **begin**
        for each *i* unlabeled **do**
              if succ(*i*)=*j*, (*i*,*j*)∈A, then
              begin
        mark node *i* labeled
        **end**

        **procedure** *Update Node Labels*
              For all *i* labeled
              **begin**
              update the vector $\left[\xi_i^h(t),\ \omega_i^h(t), \pi_i^h(t), \theta_i^h(t)\right]_{t \in T}$
                Temporary label Creation: calculate the expected time
                and variance for the newly constructed path from node i
              calculate $e_i^h(t),\ v_i^h(t)\ \ \forall t \in T$ as follows
                    $e_i^h(t) = \sum_k \left[\left(\tau_{ij}^k(t) + \left(\xi_i^h(t + \tau_{ij}^k(t)\right)\right) \cdot \rho_{ij}^k(t)\right]$
                    $\nu_i^h(t) = \sum_k \left[\rho_{ij}^k(t) * \omega_i^h\left(t + \tau_{ij}^k(t)\right)\right]$
                        $+ \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_i^h\left(t + \tau_{ij}^k(t)\right) - e_i^h(t)\right)^2\right]$
                    $\pi_i^h(t) = j, \quad \theta_i^h(t) = p$
                        where $k$ is the set of indices of possible travel
                        times on arc($i,j$) at time $t$.
                            $\xi_i^h(t) = e_i^h(t)$
                            $\omega_i^h(t) = \nu_i^h(t)$
                Label comparisons
                    Compare $\omega_i^h(t)$ with $\Omega_d(t)$, for all h
                    if $\omega_i^h(t) < \Omega_i(t)$ then
                            $\Omega_i(t) = \omega_i^h(t), \quad \pi_i^h(t) = j, \quad \theta_i^h(t) = p$
                    otherwise keep previous information
                    If $\omega_i^h(t)$ is candidate-optimal, add the path
                    information p into q(i) and put this node-path label
                    pair in the SA list.
                    Check if all $h \in q(i)$ are still candidate-optimal and
                    remove the non- candidate-optimal, labels from q(i).

                    if $(i, p) \notin SA$ then
                    put (i,p) in set SA list
            end

    Remove (j, p) from SA
            Unlabeled all nodes
    end
end
```

4.3.2 Discussion of Algorithm PMV

The PMV can be viewed as an efficient specialized modified label correcting algorithm for determining the minimum variance path from all *i* to a select destination, *d*. Similar to the Time-dependent Least-time Problem (TDLTP) algorithm of Ziliaskopoulos and Mahmassani (1993) for determining least-time paths in deterministic, time-varying networks, the PMV algorithm employs a vector label at each node, each component of which is associated with a given departure time interval. In the TDLTP algorithm, each component maintains the least time known thus far from the associated node to the destination node, for the given departure time. Similarly, each component of the vector label used in the PMV algorithm maintains the minimum variance travel time known thus far from the associated node to the destination node for the corresponding departure time. A vector label associated with node 1 is depicted in Table 4.1. In this example, the peak period consists of six time intervals (t_0 through t_5). For each departure time interval, the minimum variance of travel time (denoted by $\Omega(t)$) from this node to the destination is given in the vector label component and the associated successor node and subpath is given to the right of the component. For example, at time 3, the minimum variance path has a variance value of 2.8 units of time, and the next node of this path at time 4 is node 2 and subpath from this node 2 is 1.

Table 4.1. Example of vector label with five time intervals

Departure Time	Minimum Variance $\Omega_1(t)$	next node routing	Subpath routing
0	2.4	2	1
1	1.9	3	1
2	4.7	3	2
3	2.8	2	1

4	5.4	4	2
t>5	5.4	4	2

Lemma 4.1. The PMV algorithm terminates with the set of minimum variance of travel time paths. The following relation holds for every label at every $t \in$T:

$$\Omega_i(t) \le \sum_k \left[\rho_{ij}^k(t) * v_j^h\left(t+\tau_{ij}^k(t)\right)\right] + \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + e_j^h\left(t+\tau_{ij}^k(t)\right) - e_i^h(t)\right)^2\right], \forall t \in T,\ h \in \{1,2,\ldots,P\}$$

Proof. At the end of each iteration,

$$\Omega_i(t) = \min_{j \in A(i)} \left\{ \omega_i^h(t) = v_i^h(t) = \sum_k \left[\rho_{ij}^k(t) * v_j^h\left(t+\tau_{ij}^k(t)\right)\right] + \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + e_j^h\left(t+\tau_{ij}^k(t)\right) - e_i^h(t)\right)^2\right] \right\}$$

as required in *Update Node Labels* of the algorithm. Thus, there can be no *j* (*j=secc(i)*) such that

$$\Omega_i(t) > \sum_k \left[\rho_{ij}^k(t) * v_j^h\left(t+\tau_{ij}^k(t)\right)\right] + \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + e_j^h\left(t+\tau_{ij}^k(t)\right) - e_i^h(t)\right)^2\right].$$

Since the label components corresponding to a particular departure's time interval are permanently set once all labels at the same departure time have been determined, the proposed relation must hold.

Proposition 4.1. The PMV algorithm terminates in a finite number of steps.

Proof. The algorithm terminates in a finite number of steps if the SA list is empty in a finite number of steps. Suppose that the SA list does not get empty in a finite number of steps, then at least one node-label pair must be inserted in the SA an infinite number of times. This implies that the label at the node has improved by at least a positive real-value of travel time. If the improvement at the node continues an infinite number of times, then the variance of travel time on the path would eventually become negative, which contradicts the positive variance of travel times. This contradicts the supposition that the SA list is not empty in a finite number of steps and hence shows that the PMV algorithm terminates in a finite number of steps.

The actual number of paths that may have the minimum variance for one or more departure time intervals must be no greater than *TI*, because at most, one path has the minimum variance for each departure time (ties broken arbitrarily). However, an arbitrarily large (but finite) number of labels may need to be maintained at each node. Therefore, in a worst-case scenario, this algorithm can perform very poorly - nonpolynomially. This is shown in Proposition 4.2.

Proposition 4.2. The PMV algorithm have a worst-case computational complexity that grows exponentially with the number of nodes if $TI > 1$, where *TI* is the number of time intervals.

Proof. Assume $TI = 2$. A label for every possible path from a node to the destination node may need to be maintained, because no label may beat another label over all time intervals. Assume $\delta > 0$, then the following may occur:

	Path 1	Path 2	Path 3	Path 4	…….
Time=1	5	$5+\delta$	$5+2\delta$	$5+3\delta$	…….
Time=2	3	$3-\delta$	$3-2\delta$	$3-3\delta$	…….

No path listed above is better than any other for both time intervals. Thus, all paths must be maintained. This applies to $TI > 2$.

4.4 Algorithm for A Priori Minimum Mean-Variance path Problems: Implementation of PMV Algorithms

Since expected travel time is the primary criterion in routing optimization, and variance is secondary, it is necessary to design algorithms that minimize expected travel time and variance. In this section, we design algorithms that minimize a linear combination of expected travel time and travel time variance. Therefore, the algorithm, PMMV, is developed for a priori mean-variance path routing. Since the PMMV algorithm has very similar procedures with the algorithm PMV, the algorithm PMV is easily extended for determining a priori minimum mean-variance paths in networks where the arc travel times are random variables with time-dependent probability distribution functions. In a PMV algorithm, the node vector label $\Omega_i^h(t)$ is maintained for the variance of travel time along path *h* from node *i* to the destination at time *t*. The new vector label $\Delta_i^h(t)$ is maintained for the mean-variance combination along path h for a PMV algorithm.

4.4.1 An Algorithms

For each node i∈N and each potentially optimal path *h* to the destination node *d*, a vector label $\delta_i^h(t), t \in T$ is maintained, where $\delta_i^h(t), t \in T$ is the mean-variance combination along path *h* from node *i* to the destination, leaving node *i* at time *t*; i.e., $\delta_i^h(t) = \left\lfloor e_i^h(t) + \alpha * \sqrt{v_i^h(t)} \right\rfloor, \ \forall t \in T$. These labels are called candidate-optimal because each is potentially optimal for one or more time intervals. At each iteration of the algorithm, a node j is scanned and a temporary label vector is constructed, $\delta_i^h(t) \leftarrow \left\lfloor e_i^h(t) + \alpha * \sqrt{\nu_i^h(t)} \right\rfloor$, from each of its predecessor nodes, i∈A(i, j). This temporary label is compared with the candidate-optimal labels at node i, Δ_j (t), according to the following conditions:

$\delta_i^h(t)$ corresponds to a candidate-optimal path if $\exists$ no path $h \in q(i)$

such that $\Delta_i^h(t) \le \delta_i^h(t)$ $\forall t \in T$,otherwise the path is dominated.

This approach that we adopt will allow us to study trade-offs between mean and variance. Our route guidance model is intended to help travelers make choices that reflect their decision-making process better. A flow chart the basic procedure steps of the TAMMV2 algorithm is presented in Figure 5.4.

```
Algorithm PMMV
begin
    Procedure Initialization
        begin
            create the NODE_LIST, SA_LIST
            put all nodes i to NODE_LIST
            set SA=Ø
            set each node i,
                $\xi_i^h(t) = \infty,\ \forall i \in N-d,\ t \in T,\ h \in \{1,2,...,P\}$
                  where P is a large enough number to permit as many
                  candidate-optimal path at any node as might be required
                $\xi_d^h(t) = 0,\ \forall t \in T$
                $\omega_i^h(t) = \infty,\ \forall i \in N-d,\ t \in T,\ h \in \{1,2,...,P\}$
                $\omega_d^h(t) = 0,\ \forall t \in T$
                $\Delta_i(t) = \infty,\ \forall i \in N - d,\ t \in T,\ h \in \{1,2,...,P\}$
                $\Delta_d(t) = 0,\ \forall t \in T$
                $\pi_i^h(t) = \infty,\ \forall i \in N, t \in T,\ h \in \{1,2,...,P\}$
                $\theta_i^h(t) = \infty,\ \forall t \in T,\ h \in \{1,2,...,P\}$
                q(d) = 1 (put the first path label at node d)
            Insert destination node and path label pair (d, 1) to set SA list
        end
while SA≠Ø do
  begin
            select the first node and path label from the set SA
        call this node the current node, j
        scan the current node, j
            begin
            for each i unlabeled do
                if succ(i)=j, (i,j)∈A, then
                begin
            mark node i labeled
            end

            procedure Update Node Labels
                For all i labeled
                begin
                  update the vector $\left[\xi_i^h(t),\ \omega_i^h(t), \pi_i^h(t), \theta_i^h(t)\right]_{t\in T}$
                  calculate $e_i^h(t),\ \nu_i^h(t)\ \ \forall t \in T$ as follows
                        $e_i^h(t) = \sum_k \left[\left(\tau_{ij}^k(t) + \left(\xi_i^h(t + \tau_{ij}^k(t)\right)\right) \cdot \rho_{ij}^k(t)\right]$
                $\nu_i^h(t) = \sum_k \left[\rho_{ij}^k(t) * \omega_i^h\left(t + \tau_{ij}^k(t)\right)\right] + \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_i^h\left(t + \tau_{ij}^k(t)\right) - e_i^h(t)\right)^2\right]$

                $\pi_i^h(t) = j, \quad \theta_i^h(t) = p$

                    where $k$ is the set of indices of possible travel
                    times on arc($i,j$) at time $t$.

                        $\xi_i^h(t) = e_i^h(t)$

                        $\omega_i^h(t) = \nu_i^h(t)$

                        $\delta_i^h(t) = \left\lfloor \xi_i^h(t) + \alpha * \sqrt{\omega_i^h(t)} \right\rfloor$

                *Label comparisons*

                Compare $\delta_i^h(t)$ with $\Delta_i(t)$, for all h

                **if** $\delta_i^h(t) < \Delta_i(t)$ **then**

                        $\Delta_i(t) = \delta_i^h(t), \quad \pi_i^h(t) = j, \quad \theta_i^h(t) = p$

                **otherwise** keep previous information

            If $\delta_i^h(t)$ is candidate-optimal, then add the path information
            $p$ into $q$(i) and put this node-path label pair on the SA list.
            Check if all $h \in q$(i) are still candidate-optimal and remove the
            non- candidate-optimal labels from q(i).

                **if** $(i, p) \notin SA$ **then**

                put $(i,p)$ in set SA list

            **end**

        Remove (j, p) from SA

        Unlabeled all nodes

    **end**

**end**
```

4.4.2 Discussion of Algorithm PMMV

The PMMV also can be viewed as an efficient specialized modified label correcting algorithm for determining the minimum mean-variance path from all *i* to a select destination, *d*. Similar to the PMV algorithm, the PMMV algorithm employs a vector label at each node, each component of which is associated with a given departure time interval. In the PMV algorithm, each component maintains the least variance travel time path from the associated node to the destination node, for the given departure time. Similarly, each component of the vector label used in the PMMV algorithm maintains the minimum mean-variance travel time path from the associated node to the destination node for the corresponding departure time. Proposition 4.1 and 4.2 are also applied for the PMMV algorithm.

Lemma 4.2. The PMMV algorithm terminates with the set of minimum mean-variance of travel time paths. The following relation holds for every label at every $t \in T$:

$$\Delta_i(t) \leq \left(e_i^h(t) + \alpha * \sqrt{v_i^h(t)}\right), \quad \forall i \in N,\ h \in \{1,2,\ldots,p\},\ \forall t \in T$$

Proof. As we verify the principle of optimality of minimum expected travel time and minimum variance, we can say that the principle of optimality holds for the minimization of linear combination of mean-variance routing policy. Linear combination of above two equation is $e_i^h(t) + \alpha * \sqrt{v_i^h(t)}$. The optimality conditions are:

$$e_i^h(t) = \left(\sum_k \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_k \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right]\right), \quad \forall i \in N,\ \forall t \in T$$

$$v_i^h(t) = \left(\sum_k p_{ij}^k(t) * v_j(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t + \tau_{ij}^k(t)) - e_i(t)\right)^2\right), \ \forall i \in N,\ \forall t \in T$$

$$\delta_i^h(t) = \min_{j \in A(i)} \left(e_i^h(t) + \alpha * \sqrt{v_i^h(t)}\right), \quad \forall i \in N,\ \forall t \in T,\ h \in \{1,2,\ldots,p\}$$

At the end of each iteration,

$$\Delta_i(t) = \min_{j \in A(i)} \left\{ \begin{array}{l} \left(\sum_k \left[\tau_{ij}^k(t) * p_{ij}^k(t) \right] + \sum_k \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t) \right] \right) + \\ \alpha * \sqrt{\left(\sum_k p_{ij}^k(t) * v_j^h(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j^h(t + \tau_{ij}^k(t)) - e_j^h(t) \right)^2 \right)} \end{array} \right\}$$

as required in *Update Node Labels* of the algorithm. Thus, there can be no *j* (*j=secc(i)*) such that

$$\Delta_i(t) > \left\{ \begin{array}{l} \left(\sum_k \left[\tau_{ij}^k(t) * p_{ij}^k(t) \right] + \sum_k \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t) \right] \right) + \\ \alpha * \sqrt{\left(\sum_k p_{ij}^k(t) * v_j^h(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j^h(t + \tau_{ij}^k(t)) - e_j^h(t) \right)^2 \right)} \end{array} \right\}$$

Since the label components corresponding to a particular departure's time interval are permanently set once all labels at the same departure time have been determined, the proposed relation must hold.

4.5 Illustrative Example

In this section, both the PMV and PMMV algorithms are illustrated on the example problem. The network is shown in Figure 4.2.

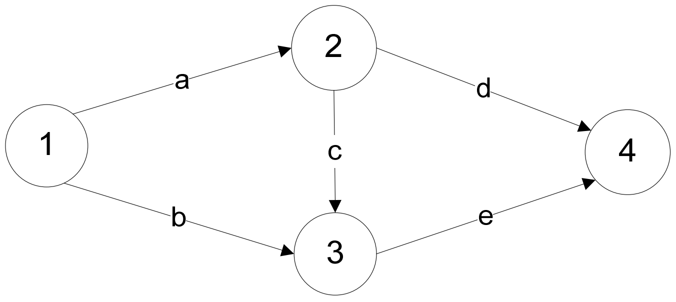


Figure 4.3. Example network

Table 4.2. Table of pmfs of travel times in Figure 4.2

Arc a		Arc b		Arc c				Arc d				Arc e							
t=0		t=0		t=2		t=3		t=2		t=3		t=4		t=5		t=6		t=7	
2	0.5	5	0.4	4	0.8	1	0.3	3	0.8	6	0.4	4	0.2	5	0.3	1	0.9	3	0.3
3	0.5	7	0.6	5	0.2	3	0.7	7	0.2	7	0.6	6	0.8	8	0.7	2	0.1	4	0.7

PMV algorithm

Initialization

Node_List N = {1,2,3,4}
Arc_List A = {a, b, c, d, e}
Time_space T = {0,1,2,3,4,5,6,7}
Destination_Node d = 4

Scan_Available List SA = ϕ

Set each node i,

$$\xi_i^h(t) = \infty, \quad \forall i \in \{1,\ 2,\ 3\},\ t \in \{0,1,2,3,4,5,6,7\},\ h \in \{1,2,3\}$$
$$\xi_4^h(t) = 0, \quad \forall\ t \in \{0,1,2,3,4,5,6,7\}, h \in \{1,2,3\}$$
$$\omega_i^h(t) = \infty, \quad \forall i \in \{1,\ 2,\ 3\},\ t \in \{0,1,2,3,4,5,6,7\},\ h \in \{1,2,3\}$$
$$\omega_4^h(t) = 0, \quad \forall\ t \in \{0,1,2,3,4,5,6,7\},\ h \in \{1,2,3\}$$
$$\Omega_i\,(t) = \infty,\ \forall i \in \{1,2,3,4\},\ t \in \{0,1,2,3,4,5,6,7\}T,\ \ h \in \{1,2,3\}$$
$$\Omega_4(t) = 0,\ \forall t \in \{0,1,2,3,4,5,6,7\},\ \ h \in \{1,2,3\}$$
$$\pi_i^h(t) = \infty,\ \ \forall\ i \in \{1,\ 2,\ 3,4\},\ t \in \{0,1,2,3,4,5,6,7\}, \mathrm{h} \in \{1,2,3\}$$
$$\theta_i^h(t) = \infty,\ \ \forall\ i \in \{1,\ 2,\ 3,4\},\ t \in \{0,1,2,3,4,5,6,7\},\ \mathrm{h} \in \{1,2,3\}$$

$q(4) = 1$

Insert the pair of destination node *d* and path label *1*, (4,1)

SA = {(d,1)}

Step 1

Select the first pair (4, 1) from the SA list
Call this is current node, *j* = 4 and path h=1

Step 2

For each *i*, $i \in A(i, j)$
i = {2, 3}

Select node 2 , i = 2, h=1
t = 2

$$e_2^1(t=2)=\left[\left(\tau_{24}^1(2)+\left(e_4^1(2+\tau_{24}^1(2)\right)\right)*\rho_{24}^1(2)\right]+\left[\left(\tau_{24}^2(2)+\left(e_4^1(2+\tau_{24}^2(2)\right)\right)*\rho_{24}^2(2)\right]$$
$$=\left(3+e_4^1(2+3)\right)*0.8+\left(7+e_4^1(2+7)\right)*0.2$$
$$=(3+0)*0.8+(7+0)*0.2$$
$$=3.8$$
$$\nu_2^1(t=2)=\left[\rho_{24}^1(2)*\nu_4^1\left(2+\tau_{24}^1(2)\right)+\rho_{24}^2(2)*\nu_4^1\left(2+\tau_{24}^2(2)\right)\right]$$
$$+\begin{bmatrix}\rho_{24}^1(2)*\left(\tau_{24}^1(2)+e_4^1(2+\tau_{24}^1(2))-e_2^1(2)\right)^2\\+\rho_{24}^2(2)*\left(\tau_{24}^2(2)+e_4^1(2+\tau_{24}^2(2))-e_2^1(2)\right)^2\end{bmatrix}$$
$$=\left[0.8*0+0.2*0\right]+\left[0.8*(3+0-3.8)^2+0.2*(7+0-3.8)^2\right]$$
$$=2.56$$

Temporary label $\xi_2^1(t=2)=e_2^1(t=2)=3.8$

$$\omega_2^1(t=2)=\nu_2^1(t=2)=2.56$$

Since $\omega_i(t=2)=2.56<\Omega_2(t=2)=\infty$,
<u>Update</u> $\Omega_2(2)$ =2.56, $\pi_2^1(2)=4$, $\theta_2^1(2)=1$

t = 3
$$e_2^1(t=3)=\left[\left(\tau_{24}^1(3)+\left(e_4^1(3+\tau_{24}^1(3)\right)\right)*\rho_{24}^1(3)\right]+\left[\left(\tau_{24}^2(3)+\left(e_4^1(3+\tau_{24}^2(3)\right)\right)*\rho_{24}^2(3)\right]$$
$$=\left(6+e_4^1(2+3)\right)*0.4+\left(7+e_4^1(2+7)\right)*0.6$$
$$=(6+0)*0.4+(7+0)*0.6$$
$$=6.6$$
$$\nu_2^1(t=3)=\left[\rho_{24}^1(3)*\nu_4^1\left(3+\tau_{24}^1(3)\right)+\rho_{24}^2(3)*\nu_4^1\left(3+\tau_{24}^2(3)\right)\right]$$
$$+\begin{bmatrix}\rho_{24}^1(3)*\left(\tau_{24}^1(3)+e_4^1(3+\tau_{24}^1(3))-e_2^1(3)\right)^2\\+\rho_{24}^2(3)*\left(\tau_{24}^2(3)+e_4^1(3+\tau_{24}^2(3))-e_2^1(3)\right)^2\end{bmatrix}$$
$$=0.24$$

Temporary label $\xi_2^1(t=3)=e_2^1(t=3)$ =6.6

$$\omega_2^1(t=3)=\nu_2^1(t=3)=0.24$$

Since $\omega_i(t=3)$ =0.24 < $\Omega_2(t=3)=\infty$,
<u>Update</u> $\Omega_2(3)$ =0.24, $\pi_2^1(3)=4$, $\theta_2^1(3)=1$

q(2)=1
If *i* is not in SA list, Put *i* and h=1in SA list, (2,1)

SA = {(2,1)}

Select node 3 , i = 3, h=1

t = 4

$e_3^1(t=4)=5.6$

$v_3^1(t=4)=0.64$

$\xi_3^1(t=4)=e_3^1(t=4)=5.6$

$\omega_3^1(t=4)=v_3^1(t=4)=0.64$

<u>Update</u> $\Omega_3(4)=0.64$, $\pi_3^1(4)=4$, $\theta_3^1(4)=1$

t = 5

$e_3^1(t=5)=7.1$

$v_3^1(t=5)=1.89$

$\xi_3^1(t=5)=e_3^1(t=5)=7.1$

$\omega_3^1(t=5)=v_3^1(t=5)=1.89$

<u>Update</u> $\Omega_3(5)=1.89$, $\pi_3^1(5)=4$, $\theta_3^1(5)=1$

t = 6

$e_3^1(t=6)=1.1$

$v_3^1(t=6)=0.09$

$\xi_3^1(t=6)=e_3^1(t=6)=1.1$

$\omega_3^1(t=6)=v_3^1(t=6)=0.09$

<u>Update</u> $\Omega_3(6)=0.09$, $\pi_3^1(6)=4$, $\theta_3^1(6)=1$

t=7

$e_3^1(t=7)=3.7$

$v_3^1(t=7)=0.21$

$\xi_3^1(t=7)=e_3^1(t=7)=3.7$

$\omega_3^1(t=7)=v_3^1(t=7)=0.21$

<u>Update</u> $\Omega_3(4)=0.21$, $\pi_3^1(4)=4$, $\theta_3^1(3)=1$

q(3)=1

If *i* is not in SA list, Put node 3 and label 1 pair in SA list

SA = {(2,1),(3,1)}

GO TO STEP 1

<u>Step 1</u>

Select the first pair (2,1) from the SA list
Call this is current pair, ***j* = 2, h=1**

<u>Step 2</u>

For each i, $(i, j) \in A$

$i =$ $\{1\}$

Select node 1, i = 1, j=2, h=1

t = 0

$e_1^1(t=0)$=7.7

$v_1^1(t=0)$=5.01

$\xi_1^1(t=0) = e_1^1(t=0)$ =7.7

$\omega_1^1(t=0) = v_1^1(t=0)$ =5.01

Since $\omega_1^1(t=0)$ =5.01 < $\Omega_1(t=0) = \infty$,

<u>Update</u> $\Omega_1(0)$ =5.01, $\pi_1^1(0) = 2$, $\theta_1^1(0) = 1$

q(1)=1

Node label pair(1,1) is not in SA list, Put (1,1)in SA list

SA = {(3,1), (1.1)}

GO TO STEP 1

<u>Step 1</u>

Select the first pair (3,1) from the SA list
Call this is current pair, ***j* = 3, h=1, SA ={(1.1)}**

<u>Step 2</u>

For each i, $(i, j) \in A$

$i =$ $\{1, 2\}$

Select node 1, i = 1, j=3, h=1

t = 0

$e_1^1(t=0) = 11.26$

$v_1^1(t=0)$=1.35

$\xi_1^2(t=0) = e_1^1(t=0)$ =11.26

$\omega_1^2(t=0) = v_1^1(t=0)$ =1.35

Since $\omega_1^2(t=0)$=1.35 < $\Omega_1(t=0)$ = 5.01

Update $\Omega_1(0)$ =1.35, $\pi_1^2(0)=3$, $\theta_1^2(0)=1$

q(1)=2

Pair (1,2) is already in SA list,

SA = {(1,1) (1,2)}

Select node 2, i = 2 (j = 3), h=1

t = 2

$e_2^1(t=2)$=5.82

$v_2^1(t=2)$=2.188

$\xi_2^2(t=2)$=$e_2^1(t=2)$ =5.82

$\omega_2^2(t=2)=v_2^1(t=2)$=2.188

Since $\omega_2^2(t=2)$=2.188 < $\Omega_2(t=2)$ = 2.56

Update $\Omega_2(2)$ =2.188, $\pi_2^2(2)=3$, $\theta_2^2(2)=2$

t = 3

$e_2^1(t=3)$=4.85

$v_2^1(t=3)$=1.568

$\xi_2^2(t=3)$=$e_2^1(t=3)$ =4.85

$\omega_2^2(t=3)=v_2^1(t=3)$=1.568

Since $\omega_2^2(t=3)$ =1.568>$\Omega_2(t=3)=0.24$

Keep $\Omega_2(3)$ =0.24, $\pi_2^1(3)=4$, $\theta_2^1(3)=1$

node 1 is available and h=3, pair (1,2) in SA list, **SA = {(1,1) (1,2), (2, 2)}**

q(2)=2

GO TO STEP 1

Select the first pair (1,1) from the SA list

Call this is current node, ***j* = 1, SA ={(2,2)}**

Step 2

For each i, $(i,j)\in A$

$i=\ \ \phi$

GO TO STEP 1

Select the first pair (1,2) from the SA list
Call this is current node, ***j* = 1, SA ={(2,2)}**

Step 2

For each i, $(i, j) \in A$
$i = \quad \phi$

GO TO STEP 1

Select the pair (2,2) from the SA list
Call this is current node, ***j* = 2, SA ={ }, h=2**

Select node 1, i = 1
t = 0
$e_1^2(t=0)$=6.825
$v_1^2(t=0)$=5.6
$\xi_1^3(t=0)=e_1^2(t=0)$=4.85
$\omega_1^3(t=0)=v_1^2(t=0)$=1.568

Since $\omega_1^3(t=0)$ =5.6 > $\Omega_1(t=0)=$ 1.35
Keep $\Omega_1(0)$ =1.35, $\pi_1^3(0)=3$, $\theta_1^3(0)=2$

SA = { }

STOP

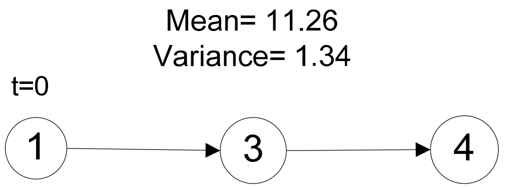


Figure 4.4. Resulting minimum variance path for departure time 0 at node1 for the example problem

PMMV algorithm

Node 2

j=4, h=1
t=2
Temporary label $\xi_2^1(t=2) = 3.8$

$\omega_2^1(t=2) = 2.56$

$\delta_i^h(t=2) = \lfloor 3.8 + 1 * \sqrt{2.56} \rfloor = 5.4$

Since $\delta_2^1(t=2) = 5.4 < \Delta_2(t=2) = \infty$

Update $\Delta_2(2)$ =5.4, $\xi_2^1(2) = 3.8$ $\omega_2^1(2) = 2.56$ $\pi_2^1(2) = 4$, $\theta_2^1(2) = 1$

t = 3
Temporary label $\xi_2^1(t=3)$ =6.6

$\omega_2^1(t=3)$ =0.24

$\delta_2^1(t=3) = \lfloor 6.6 + 1 * \sqrt{0.24} \rfloor = 7.1$

Since $\delta_2^1(t=3) = 7.1 < \Delta_2(t=3) = \infty$

Update $\Delta_2(3)$ =7.1, $\xi_2^1(3) = 3.8$ $\omega_2^1(3) = 2.56$ $\pi_2^1(3) = 4$, $\theta_2^1(3) = 1$

Node 3

j=4, h=1
t = 4

$\xi_3^1(t=4) = e_3^1(t=4)$ =5.6

$\omega_3^1(t=4) = v_3^1(t=4)$ =0.64

$\delta_3^1(t=4) = \lfloor 5.6 + 1 * \sqrt{0.64} \rfloor = 6.4$

Update $\Delta_3(4)$ =6.4, $\pi_3^1(4) = 4$, $\theta_3^1(4) = 1$

t = 5

$\xi_3^1(t=5) = e_3^1(t=5)$ =7.1

$\omega_3^1(t=5) = v_3^1(t=5)$ =1.89

$\delta_3^1(t=5) = \lfloor 7.1 + 1 * \sqrt{1.89} \rfloor = 8.44$

Update $\Delta_3(5)$ =8.44, $\pi_3^1(5) = 4$, $\theta_3^1(5) = 1$

t = 6

$\xi_3^1(t=6) = e_3^1(t=6)$ =1.1

$\omega_3^1(t=6) = v_3^1(t=6)$ =0.09

$\delta_3^1(t=6) = \lfloor 1.1 + 1 * \sqrt{0.09} \rfloor = 1.4$

Update $\Delta_3(6)$ =0.09, $\pi_3^1(6) = 4$, $\theta_3^1(6) = 1$

t=7

$\xi_3^1(t=7)=e_3^1(t=7)$ =3.7

$\omega_3^1(t=7)=\nu_3^1(t=7)$ =0.21

$\delta_3^1(t=7)=\left\lfloor 3.7+1*\sqrt{0.21}\right\rfloor=4.16$

Update $\Delta_3(4)$ =4.16, $\pi_3^1(4)=4$, $\theta_3^1(3)=1$

Node 1

j=2, h=1

t = 0

$\xi_1^1(t=0)=e_1^1(t=0)$ =7.7

$\omega_1^1(t=0)=\nu_1^1(t=0)$ =5.01

$\delta_1^1(t=0)=\left\lfloor 7.7+1*\sqrt{5.01}\right\rfloor=9.94$

Since $\omega_1^1(t=0)$ =9.94 < $\Omega_1(t=0)=\infty$,

Update $\Delta_1(0)$ =9.94, $\pi_1^1(0)=2$, $\theta_1^1(0)=1$

Node 1

j=3, h=1

t = 0

$\xi_1^2(t=0)=e_1^1(t=0)$ =11.26

$\omega_1^2(t=0)=\nu_1^1(t=0)$ =1.35

$\delta_1^2(t=0)=\left\lfloor 11.26+1*\sqrt{1.35}\right\rfloor$ =12.16

Since $\delta_1^2(t=0)$=12.16 > $\Delta_1(t=0)$ = 9.94

Keep $\Delta_1(0)$ =9.94, $\pi_1^1(0)=2$, $\theta_1^1(0)=1$

Node 2

j=3, h=1

t=2

Temporary label $\xi_2^2(t=2)=e_2^2(t=2)$ =5.82

$\omega_2^2(t=2)=\nu_2^2(t=2)$ =2.188

$\delta_2^2(t=2)=\left\lfloor 5.82+1*\sqrt{2.188}\right\rfloor$ = 7.30

Since $\delta_2^2(t=2)$=7.30 > $\Delta_2(t=2)$ = 5.4

Keep $\Delta_2(2)$ =5.4, $\pi_2^1(2)=4$, $\theta_2^1(2)=1$

t = 3

Temporary label $\xi_2^2(t=3)$ =4.85

$\omega_2^2(t=3)$ =1.568

$\delta_2^2(t=3)=\left\lfloor 4.85+1*\sqrt{1.568}\right\rfloor$ = 6.10

Since $\delta_2^2(t=3) = 6.10 < \Delta_2(t=3)$ =7.1

Update $\Delta_2(3)$ =6.1, $\xi_2^2(3) = 4.85$ $\omega_2^2(3) = 1.568$ $\pi_2^2(3) = 3$, $\theta_2^1(3) = 1$

Node 1

j=2, h=2

t = 0

$\xi_1^3(t=0) = e_1^2(t=0)$ =6.825

$\omega_1^3(t=0) = \nu_1^2(t=0)$ =5.6

$\delta_1^3(t=0) = \lfloor 6.825 + 1 * \sqrt{5.6} \rfloor$ =9.19

Since $\delta_1^3(t=0)$ =9.19 < $\Delta_1(t=0)$ = 9.94

Update $\Delta_1(0)$ =9.94, $\pi_1^3(0) = 2$, $\theta_1^3(0) = 2$

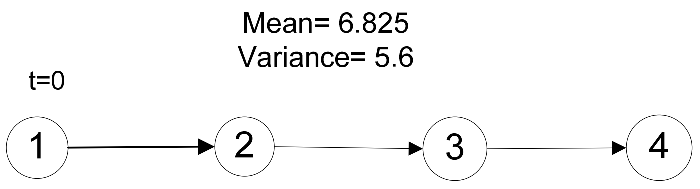


Figure 4.5. Resulting minimum mean-variance path for departure time 0 at node1 for the example problem

4.6 Concluding Remarks

In this chapter, two specialized modified label correcting algorithms are presented for generating all of the a priori minimum variance paths and minimum mean-variance paths, and from all nodes to a single destination for all departure times in the peak period in a network with stochastic, time-dependent arc times.

Both PMV and PMMV algorithms have nonpolynomial worst-case complexity because the number of labels required determining a priori minimum variance paths may grow exponentially with the size of the network.

The two algorithms are complicated by the need to maintain the minimum variance or minimum mean-variance values for every time interval of every path. Intuitively, it seems possible that only the best label for each departure time interval, with individual pointers for each, needs to be maintained, as required of the deterministic, time-dependent shortest path problem (Ziliaskopoulos and Mahmassani, 1993). Unfortunately, if the two labels are compared, one must dominate the other over all departure times in the peak period in order to eliminate a path from future consideration. In order to maintain the path information, one must keep a separate label for each path that is non-dominated for at least one departure time. This is because it is possible to arrive at the origin of a subpath in more than one time interval, and therefore, the subpath's labels at more than one time interval may contribute to the construction of the label of the path containing this subpath. Only upon termination of the algorithm can the non-dominated optimal paths, or optimal paths, be evaluated at specific time intervals; and thus, for each time interval, the number of non-dominated solutions may be smaller than is found for the entire time period.

The two algorithms are presented specifically for a priori path selection, assuming that the user is unable to dynamically change course en route. In reality, as a vehicle travels along a selected route, the travel time from the origin to the current location of the vehicle is no longer uncertain because it has already occurred. In Chapter 5, the PMMV algorithms are extended for generating the paths for use in a time-adaptive route choice framework where the vehicle selects the path on which it would continue based on the revealed (actual) arrival time at each node.

Chapter 5. Time-Adaptive Minimum Mean-Variance Algorithms

In both transportation and data communication systems, stochastic time-dependent network provides a more realistic representation of actual travel conditions on which to base critical routing decisions (e.g., for emergency response or priority data transfers) than commonly used deterministic or static models. Optimal routes with respect to deterministic network attributes (such as distance) may be chosen before travel begins, because the optimal path on which to continue does not change as a motorist or packet traverses the network. However, in transportation and data networks, where future arc traversal times are uncertain and conditions are changing over time, one can make improved routing decisions en route as travel times on traveled arcs are revealed. The selection of a route prior to travel is referred to as *a priori best path routing*, because it is assumed that the path is chosen in its entirety before travel begins. The selection of a route where arc traversal times are revealed en route once the arc is traversed can be viewed as a multistage recourse problem, where recourse decisions can be taken in response to realizations of arc traversal time outcomes that are not known a priori. This type of route selection is the focus of this Chapter and is referred as *"best" next arc routing* or *time-adaptive routing*.

In this context, for a given origin-destination pair at a specific departure time, a single path may not provide an adequate solution, because the optimal path depends on intermediate information concerning experienced travel times on traveled arcs. Therefore, attention should be focused on finding a set of 'non-dominated' or 'efficient' routes.

Furthermore, travel time measures for routing are highly uncertain. When the measures are uncertain, choosing among routes becomes even more difficult because the tradeoffs among measures are less precise - the decision maker is forced to choose one probability distribution over another, rather than choosing one value over another. A method for comparing probability distributions which is less strict than the classical stochastic dominance was presented in section 5.1. This method has important practical application in determining a set of non-dominated routes in a network when there are multiple, uncertain measures which form the basis for route evaluation.

In section 5.2, two efficient algorithms, TAMMV1 and TAMMV2, are described for determining time-adaptive minimum mean-variance routing in stochastic time-dependent networks. We design algorithms that minimize a linear combination of mean and variance of travel times from origin to destination. In the previous chapter, it provides a theoretical base for the algorithms in section 5.2. Concluding remarks are given in section 4.6. Our route guidance model is intended to help travelers make choices that reflect their decision-making process better.

5.1 Non-Dominated Path Selection for Mean-Variance Routing

In this section, path comparisons for a priori and time-adaptive decisions in stochastic, time-dependent networks are studied, and a method for comparing probability distributions which is less strict than the classical stochastic dominance is described. The method includes a probability parameter which permits control of the degree to which the comparison deviates from the classical stochastic dominance. This method has important practical application in determining a set of non-dominated routes in a network when there are multiple, uncertain measures which form the basis for route evaluation.

Consider paths starting at node *i* and ending at node *d*. A path attribute L_i^k is the summation of link attributes along the path R_i^k where k is the path index. Therefore,

$L_i^k = l_{im} + ... + l_{nd}$

where $l_{im},..., l_{nd}$ designate the representative attributes of links (i, m),...,(n, d) belonging to path R_i^k. Hence, L_i^k is also a random variable with mean μ_i^k and variance ν_i^k. The following criterion is proposed to choose one path over another.

Consider two paths R_i^1 and R_i^2 from node *i* to destination node. (Note: when several paths to the same departure node are considered, the subscript *i* will be dropped for simplicity in notation)

> Path R^1 is preferred to path R^2 if attribute L^1 is stochastically (in distribution) smaller than L^2 (i.e., $L^1 < L^2$).
>
> Path L^1 is indifferent to path L^2 if L^1 is not stochastically smaller than L^2, and L^2 is not stochastically smaller than L^1 (i.e., $L^1 < L^2$ and $L^1 > L^2$).

The comparison of path attributes L^1 and L^2 can be carried out in two stages.

5.1.1 Primary Comparison Rule

Let random variable L_i^1 denote the travel time from node *i* to destination node *d* using path 1. The expected value and variance of L_i^1 are denoted as μ_i^1 and ν_i^1 respectively. L_i^2 also denotes the travel time from node *i* to destination node *d* using path 2 with expected value μ_i^2 and variance v_i^2, respectively.

It is claimed that (approximately) $L_i^1 < L_i^2$ if

$$\mu_i^1 < \mu_i^2 \text{ and } v_i^1 \le v_i^2 \tag{5-1}$$

or

$$\mu_i^1 \le \mu_i^2 \text{ and } v_i^1 < v_i^2 \tag{5-2}$$

The validity of this approximate comparison rule for path attributes should be examined based on the classical definition for stochastic comparison of random variables.

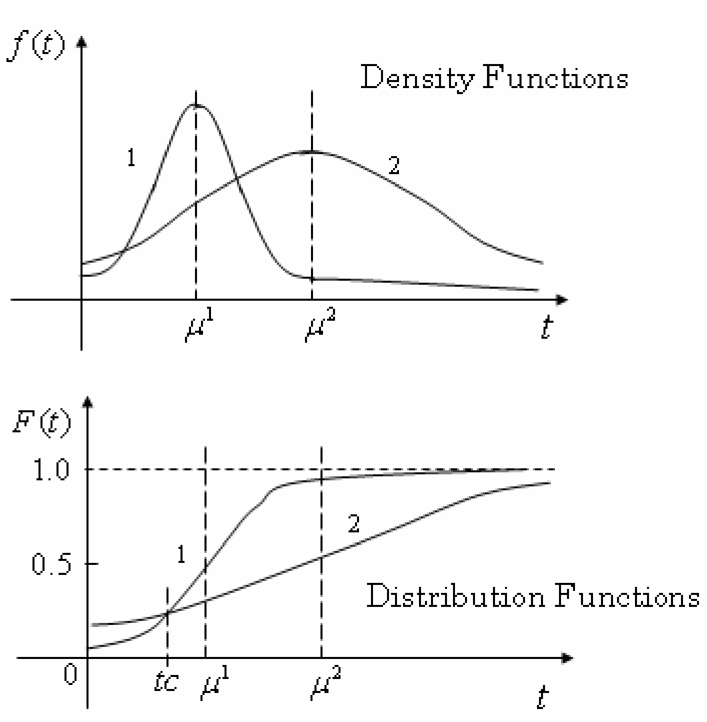


Figure 5.1. Comparison of stochastic path attributes L_i^1 and L_i^2 where $\mu_i^1 < \mu_i^2$ and $v_i^1 < v_i^2$

As shown in Figure 5.1, there is some value of t, denoted t_c, where the cumulative distribution functions cross. That is:

$$F_{L^2}(t) > F_{L^1}(t) \text{ for } t<t_c$$

and

$$F_{L^1}(t) > F_{L^2}(t) \quad \text{for } t > t_c$$

For values of t greater than t$_c$, the probability that path attribute L^1 is less than t exceeds the probability that path attribute L^2 is less than t. Also, if conditions (3) or (4) hold, the following can be observed:

$$t_c < \mu_i^1 < \mu_i^2, \qquad (5\text{-}3)$$

$$F_{L^1}(t_c) = F_{L^2}(t_c) \leq 0.5 \qquad (5\text{-}4)$$

Therefore, the approximate rule would choose paths which may not satisfy the classical definition of stochastic dominance for comparison of random variables at small values of the path attribute (t < t$_c$). In applications to hazardous materials routing, this is not a significant issue since we are mainly concerned about the possible realization of large values of the path attribute.

The primary comparison rule can be implemented using a multiobjective shortest path algorithm. Each stochastic link attribute l_{ij} is transformed to two deterministic attributes μ_{ij} and v_{ij}. Then the multiobjective algorithm produces a set of non-dominated paths from node *j* to the destination node. Paths belonging to the set S$_j$ are paths that do not satisfy conditions (5-1) or (5-2). That is, suppose the set *S$_j$*, contains the *r* non-dominated paths:

$$S_j = \{R^1, R^2, R^3, \ldots, R^r\}$$

If these paths are ordered by the increasing mean value,

$$\mu^1 < \mu^2 < \mu^3 \ldots\ldots < \mu^r$$

then, they must also be ordered in decreasing variance:

$$v^1 > v^2 > v^3 \ldots\ldots > v^r$$

Equations (5-1) and (5-2), which form the basis the primary comparison rule, allow some choices to be made among alternative routes. Routes which are clearly dominated in both mean and variance of attribute distributions are discarded by the primary comparison rule. However, through closer examination of the attribute distributions for the remaining set of routes, S_j, we may be able to further reduce the set of non-dominated routes.

5.1.2 Secondary Comparison Rule

Consider any two routes R^1 and R^2 from the set S_j with $\mu^1 < \mu^2$ and $\nu^1 > \nu^2$. As shown in Figure 5.2, there is some t, denoted t_c, where the cumulative distribution functions cross. That is,

$$F_{L^2}(t) > F_{L^1}(t) \text{ for all values of } t<t_c \qquad (5\text{-}5)$$

We can also see that

$$t_c > \mu^2 > \mu^1 \qquad (5\text{-}6)$$

and

$$F_{L^1}(t_c) = F_{L^2}(t_c) > 0.5 \qquad (5\text{-}7)$$

That is, t_c is greater than the larger mean, and the probability that either of the path attributes will take a value greater than t_c is less than 0.5. Therefore, L^1, is stochastically smaller than L^2 for 'most' values of the attribute (i.e., with probability greater than 0.5). The question addressed by the secondary comparison rule is whether or not the range in which L^1, is smaller than L^2 is large enough that we can conclude that route 1 should be preferred to route 2.

We propose the following comparison rule. Since we know

$$F_{L^1}(t) > F_{L^2}(t) \text{ for all } t<t_c,$$

and if the value of t_c is such that

$$F_{L^1}(t_c) = F_{L^2}(t_c) > 1-\alpha,$$

then we will prefer R^1 to R^2 ($L^1 < L^2$). This concept is illustrated in Figure 5.2.

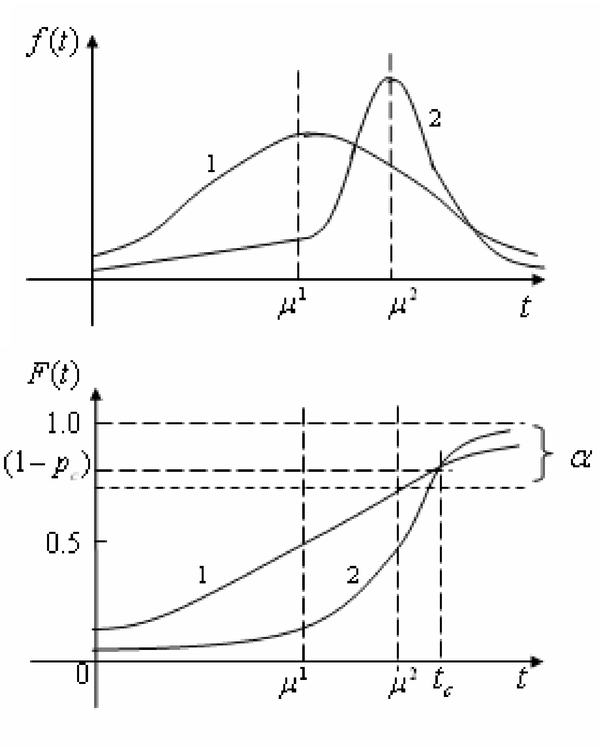


Figure 5.2. Comparison of two path travel time L_i^1 and L_i^2 where $\mu_i^1 < \mu_i^2$ and $\nu_i^1 > \nu_i^2$

The distribution of L^1 is preferred (for the purpose of choosing routes) to the distribution of L^2 as long as the classical comparison rule is satisfied for all values of t, except extremely large values which have a small probability of exceedance. This probability is controlled by specifying a small value for α.

The value of α will determine the level of error in the path attribute comparison. As we allow α to increase, we are accepting a larger probability of error in the comparison of stochastic path attributes. Since α is a parameter that is specified by the analyst, the secondary comparison rule can be used to compare paths based on the level of accuracy necessary for a specific routing problem. As α is made smaller, the secondary comparison rule becomes less powerful as a means of differentiating among alternative paths because we are insisting on a higher level of conformance to the classical notion of stochastic dominance. As α is allowed to increase, the secondary comparison rule becomes efficient at reducing the size of the possible solution set, but may discard a path which would have been of interest.

5.1.3 Algorithm TAMMV-ND

This algorithm is a specialized modified label-correcting algorithm for generating non-dominated minimum mean-variance routing. Because of the secondary comparison rule, this algorithm is not efficient for computations. Two efficient algorithms are developed in the next section. The detailed description of this algorithm is presented in the next section.

```
begin
    Procedure Initialization
        begin
            create the NODE_LIST, SA_LIST
            put all nodes i to NODE_LIST
            set SA_LIST=Ø
            set each node i,
                $\xi_i(t) = \infty,\ \forall i \in N - d,\ t \in T$
                $\xi_d(t) = 0,\ \forall t \in T$
                $\omega_i(t) = \infty,\ \forall i \in N - d,\ t \in T$
                $\omega_d(t) = 0,\ \forall t \in T$
                $\pi_i(t) = \infty,\ \forall i \in N - d, t \in T$
                $\pi_d(t) = \phi,\ \forall t \in T$
            Insert destination node, d to set SA_LIST
        end
```

```
while SA_LIST≠Ø do
begin
        select the first node of the set SA_LIST
        call this node the current node, j
        scan the current node, j
        begin
        for each i unlabeled do
                if succ(i)=j, (i,j)∈A, then
                begin
        mark node i labeled
        end

        procedure Update Node Labels
                For all i labeled
                begin
                        update the vector $[\xi_i(t),\ \omega_i(t), \pi_i(t)]_{t\in T}$
                        calculate $e_i(t),\ v_i(t)\ \ \forall t \in T$ as follows

                        $e_i(t) = \sum_k \left[\left(\tau_{ij}^k(t) + \left(\xi_j(t+\tau_{ij}^k(t)\right)\right)\cdot \rho_{ij}^k(t)\right]$
                        $v_i(t) = \sum_k \left[\rho_{ij}^k(t) * \omega_j\left(t+\tau_{ij}^k(t)\right)\right]$
                                $+ \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_j\left(t+\tau_{ij}^k(t)\right) - e_i(t)\right)^2\right]$

                        where k is the set of indices of possible travel times
                        on arc(i,j) at time t.

                        if $e_i(t) < \xi_i(t)$ and $\upsilon_i(t) < \omega_i(t)$ then
                                $\xi_i(t) = e_i(t),\ \omega_i(t) = \upsilon_i(t),\ \pi_i(t) = j$

                        if [ $e_i(t) < \xi_i(t)$ and $\upsilon_i(t) > \omega_i(t)$ ] or
                                [ $e_i(t) > \xi_i(t)$ and $\upsilon_i(t) < \omega_i(t)$ ] then
                        apply secondary comparison rule

                        otherwise keep all paths as a nondominated path

                        if $i \notin SA\_LIST$ then
                        put i in SA_LIST
                end

end
```

Secondary comparison rule

This secondary comparison between paths h$_1$ and h$_2$ is done for which $\mu^{h_1} < \mu^{h_2}$, but $\nu^{h_1} > \nu^{h_2}$ (or $\mu^{h_1} > \mu^{h_2}$, but $\nu^{h_1} < \nu^{h_2}$). Because path h$_1$ has a smaller mean value than does path h$_2$, but a larger variance, the two path specific cumulative distribution functions intersect each other. As a result, under the stochastic dominance, it is not possible to say that $\mu^{h_1} < \mu^{h_2}$. However, if the difference in the mean values is relatively large and the difference in the variances is relatively small, one might still be willing to assert a preference for path h$_1$ over path h$_2$, even though neither the mean–variance comparison nor the stochastic dominance is satisfied. This can be formalized by saying that the CDF's of path h$_1$ and h$_2$ are equal at time t$_c$ (i.e., $F_{L^1}(t_c) = F_{L^2}(t_c)$), and t$_c$ is large enough so that $F_{L^1}(t_c) = F_{L^2}(t_c) > 1-\alpha$, then we can assert a preference for path h$_1$ over path h$_2$. The quantity $\boldsymbol{\alpha}$ is a probability parameter that controls the degree to which the comparison deviates from the stochastic dominance. That is, as $\alpha \to 0$, this comparison rule converges to a standard stochastic dominance comparison, but for values of $\alpha > 0$, it is a relaxation.

5.1.4 Illustrative Example

In this section, the TAMMV-ND algorithm is illustrated on an example network shown in Figure 4.2. Since the initialization procedures are same as previous algorithm PMV, we present only the major part of the algorithm.

Initialization

Node_List N = {1,2,3,4}
Arc_List A = {a, b, c, d, e}
Time_space T = {0,1,2,3,4,5,6,7}
Destination_Node d = 4

Scan_Available List SA_LIST = ϕ

Set each node i,

$\xi_i(t) = \infty, \quad \forall i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\xi_4(t) = 0, \quad \forall\ t \in \{0,1,2,3,4,5,6,7\}$

$\omega_i(t) = \infty, \quad \forall i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\omega_4(t) = 0, \quad \forall\ t \in \{0,1,2,3,4,5,6,7\}$

$\pi_i(t) = \infty, \quad \forall\ i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\pi_4(t) = \phi, \quad \forall\ t \in \{0,1,2,3,4,5,6,7\}$

Create the Scan_Avabable list, SA_LIST, and insert the destination node d

SA_LIST = {4}

<u>Step 1</u>

Select the first node 4 from the SA_LIST list
Call this is current node, j = 4,

<u>Step 2</u>

For each i, $(i, j) \in A$

$i =$ {2, 3}

Select node 2 , i = 2, h=1

t = 2

$e_2^1(t=2)$=3.8

$v_2^1(t=2)$=2.56

Update Label

If $e_i^h(t) < \xi_i^h(t),\ and\ v_i^h(t) < \omega_i^h(t)$

Then $\xi_i^h(t) = e_i^h(t),\ and\ \omega_i^h(t) = v_i^h(t), and\ \pi_i^h(t) = (i, j)$

Since $e_2^1(t=2) = 3.8 < \xi_2^1(t=2) = \infty$,

$v_2^1(t=2) = 2.56 < \omega_2^1(t=2) = \infty$

Update $\xi_2^1(2) = 3.8$, $\omega_2^1(2) = 2.56$, $\pi_2^1(2) = d$, $h = 1$

t = 3

$e_2^1(t=3)$=6.6

$\nu_2(t=3)$=0.24

Since $e_2^1(t=3) = 6.6 < \xi_2^1(t=3) = \infty$,

$\nu_2^1(t=3) = 0.24 < \omega_2^1(t=3) = \infty$

Update $\xi_2^1(3) = 6.6$, $\omega_2^1(3) = 0.24$, $\pi_2^1(3) = d$ $h = 1$

If *i* is not in SA_LIST list, Put *i* in SA_LIST list

SA_LIST = {2}

Select node 3 , i = 3, h=1

t = 4

$$e_3^1(t=4) = \left[\left(\tau_{34}^1(4) + \left(\xi_4^1(4+\tau_{34}^1(4))\right) * \rho_{34}^1(4)\right] + \left[\left(\tau_{34}^2(4) + \left(\xi_4^1(4+\tau_{34}^2(4))\right) * \rho_{34}^2(4)\right]$$

$$= \left(4 + \xi_4^1(2+4)\right) * 0.2 + \left(6 + \xi_4^1(2+6)\right) * 0.8$$

$$= (4+0) * 0.2 + (6+0) * 0.8$$

$$= 5.6$$

$$\nu_3^1(t=4) = \left[0.2*0 + 0.8*0\right] + \left[0.2*(4+0-5.6)^2 + 0.8*(6+0-5.6)^2\right]$$

$$= 0.64$$

Since $e_3^1(t=4) = 6.6 < \xi_2^1(t=4) = \infty$

$\nu_3^1(t=4) = 0.64 < \omega_3^1(t=4) = \infty$

Update $\xi_3^1(4) = 5.6$, $\omega_3^1(4) = 0.64$, $\pi_3^1(4) = e$

t = 5

$e_3^1(t=5) = 7.1$

$\nu_3^1(t=5) = 1.89$

Since $e_3^1(t=5) = 6.6 < \xi_2^1(t=5) = \infty$

$\nu_3^1(t=5) = 1.89 < \omega_3^1(t=5) = \infty$

Update $\xi_3^1(5)$ =7.1, $\omega_3^1(5)$ =1.89, $\pi_3^1(5) = e$

t = 6

$e_3^1(t=6) = 1.1 < \xi_2^1(t=6) = \infty$

$\nu_3^1(t=6) = 0.09$

Since $\nu_3^1(t=6)$ =0.09 $<$ $\omega_3^1(t=6) = \infty$

Update $\xi_3^1(6)$ =1.1, $\omega_3^1(6)$ =0.09, $\pi_3^1(6) = e$

t = 7

$e_3^1(t=7) = 6.6 < \xi_2^1(t=7) = \infty$

$v_3^1(t=7)$= 0.21

Since $v_3^1(t=7)$ =0.21 < $\omega_3^1(t=7)=\infty$

Update $\xi_3^1(7)$ =3.7, $\omega_3^1(7)$ =0.21, $\pi_3^1(7)=e$

If *i* is not in SA_LIST, Put node 3 in SA_LIST

SA_LIST = {2, 3}

GO TO STEP 1

Select the first node 2 from the SA_LIST
Call this is current node, ***j* = 2, SA_LIST ={3}**

<u>Step 2</u>

For each i, $(i,j)\in A$

$i=\{1\}$

Select node 1, i = 1, h=1

t = 0

$e_1^1(t=0)$=7.7

$v_1^1(t=0)$=5.01

Since $e_1^1(t=0)=7.7<\xi_1^1(t=2)=\infty$,

$v_1^1(t=0)$ =5.01 < $\omega_1^1(t=0)=\infty$

Update $\xi_1^1(0)$ **=7.7,** $\omega_1^1(0)$ =5.01**,** $\pi_1^1(0)=a$

Node 1 is not in SA_LIST, Put *node 1* in SA_LIST

SA_LIST = {3,1}

GO TO STEP 1

Select the first node 3 from the SA
Call this is current node, ***j* = 3, SA_LIST ={1}**

<u>Step 2</u>

For each i, $(i,j)\in A$

$i=\{1,2\}$

Select node 1, i = 1,

t = 0

$e_1^1(t=0)$=11.26

$v_1^1(t=0)$=1.3524

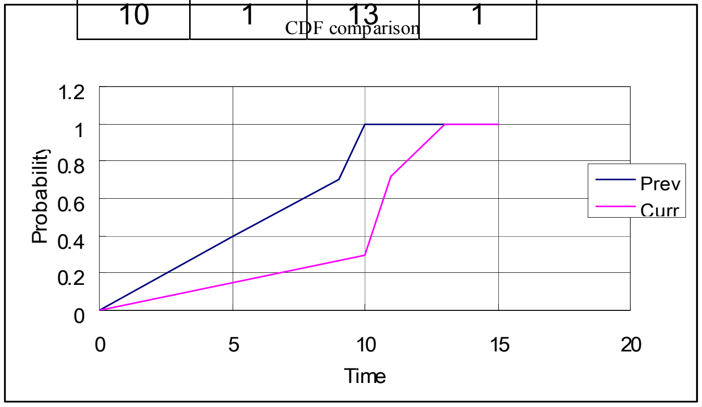
Since $e_1^1(t=0) = 11.26 > \xi_1^1(t=0) = 7.7$,

$v_1^1(t=0) = 1.35 < \omega_1^1(t=0) = 5.01$

➔ Apply secondary comparison rule

CDF calculation

Path (Prev)		Path(Curr)	
time	prob	time	prob
5	0.4	10	0.3
9	0.7	11	0.72
10	1	13	1



Find t_c, where $F(e_1^1(t_c)) = F(\xi_1^1(t_c))$

t_c is large enough, this case t_c is around 10.

Also, if α=0.5, $F(e_1^1(t_c)) = F(\xi_1^1(t_c)) > 1 - 0.5 = 0.5$

Therefore, Keep previous node label

$$\xi_1^1(0) = 7.7, \omega_1^1(0) = 5.01, \pi_1^1(0) = b$$

Node 1 is already in SA_LIST ,

SA_LIST = {3}

Select node 2, i = 2 (j = 3)

<u>t = 2</u>

$e_2^1(t=2)$=5.82

$v_2^1(t=2)$=2.188

Update Label

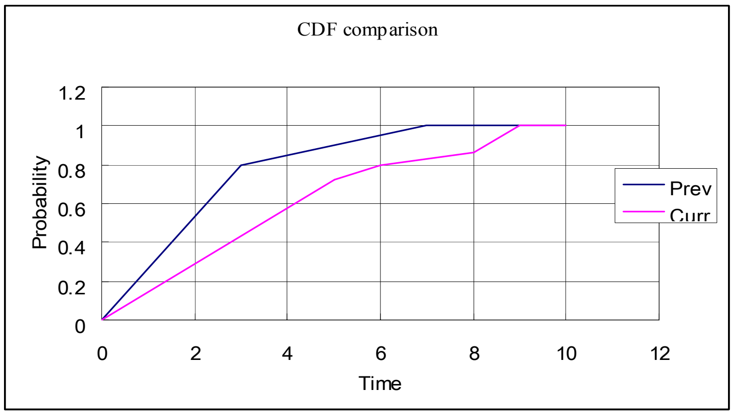
Since $e_2^1(t=2) = 5.82 > \xi_2^1(t=0) = 3.8$,

$v_2^1(t=2) = 2.188 < \omega_2^1(t=2) = 2.56$

➔ **Apply secondary comparison rule**

CDF calculation

Path (Prev)		Path(Curr)	
time	prob	time	prob
5	0.4	5	0.72
9	0.7	6	0.8
10	1	8	0.86
		9	1



Find t_c, where $F\left(e_1^1(t_c)\right) = F\left(\xi_1^1(t_c)\right)$

t_c is large enough, this case t_c is around 9.

Also, if $\alpha=0.5$, $F\left(e_1^1(t_c)\right) = F\left(\xi_1^1(t_c)\right) > 1-0.5=0.5$

Therefore, Keep previous node label

$$\xi_2^1(2)=3.8, \omega_2^1(2)=2.56,\ \pi_2^1(2)=d$$

<u>t = 3</u>

$e_2^1(t=3)$=4.85

$v_2^1(t=3)$=1.568

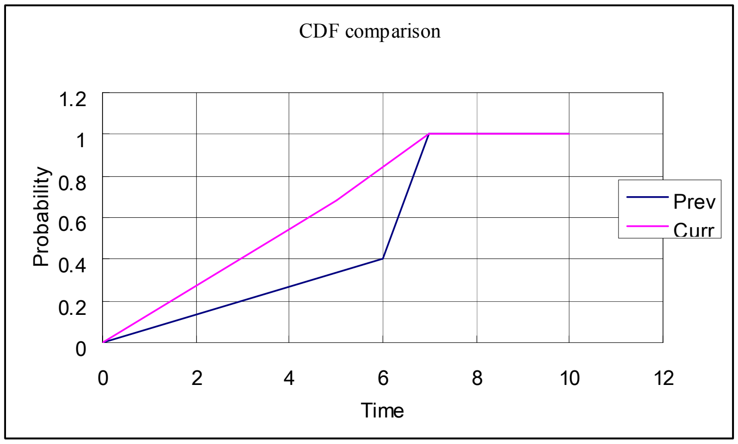
Since $e_2^1(t=3) = 4.85 < \xi_2^1(t=3)=6.6$,

$v_2^1(t=3) = 1.568 > \omega_2^1(t=3) = 0.24$

➔ **Apply secondary comparison rule**

Path (Prev)	Path(Curr)

time	prob	time	prob
6	0.4	5	0.72
7	0.7	6	0.8
10	1	8	0.86
		9	1



Find t$_c$, where $F\left(e_1^1(t_c)\right) = F\left(\xi_1^1(t_c)\right)$
t$_c$ is large enough, this case t$_c$ is around 7.
Also, if α=0.5, $F\left(e_1^1(t_c)\right) = F\left(\xi_1^1(t_c)\right) > 1-0.5 = 0.5$

Therefore, **Update** $\xi_2^1(3) = 4.85, \omega_2^1(3) = 1.568, \pi_2^1(3) = c$

If *i* is not in SA_LIST, Put node 2 in SA_LIST
SA_LIST = {1, 2}

GO TO STEP 1

Select the first node 1 from the SA_LIST
Call this is current node, ***j* = 1, SA_LIST ={2}**

<u>Step 2</u>

For each *i*, $(i, j) \in A$
$i = \quad \phi$

GO TO STEP 1

Select the first node 2 from the SA_LIST

Call this is current node, ***j* = 2, SA_LIST ={ }**

Select node 1, i = 1

t = 0

$e_1^1(t=0)$=6.825

$\nu_1^1(t=0)$=3.1146

Since $e_1^1(t=0) = 6.825 < \xi_1^1(t=0) = 7.7$

$\nu_1^1(t=0)$ =3.115 < $\omega_1^1(t=0) = 5.01$

Update $\xi_1^1(0)$ **=6.825,** $\omega_1^1(0)$ **=3.115,** $\pi_1^1(0)$ **=** ***b***

SA_LIST = { }

STOP

5.2 Time-Adaptive Mean-Variance Algorithms

As we know, we treat the link travel time as a random variable; the trip time from one node to another is also a random variable. It is assumed that travel times on the network can be treated as a multivariate random variable for which links may have correlated travel-times. Our goal is to devise an algorithm which allows a traveler to select a route by examining the mean and variance of travel time. We present an algorithm for selecting the "best" next arc routing using two comparison rules. The first comparison rule is described in the primary comparison rule from the previous chapter. The second comparison rule is a linear combination of relevant attributes of a routing policy. The problem of optimal routing policy with minimum mean and minimum variance (from the first comparison rule) and minimum linear combination of expected travel time and variance (from second comparison rule) problem in a stochastic time-dependent network with one destination node *d* is to find $\pi_i(t)$ such that,

$$\pi_i(t) = Arg\min_{j\in A(i)} \left(\xi_i(t),\, \omega_i(t)\right), \quad \forall i \in N,\, t \in T$$

or

$$\pi_i(t) = Arg\min_{j\in A(i)} \left(\xi_i(t) + \alpha * \sqrt{\omega_i(t)}\right), \quad \forall i \in N,\, t \in T$$

The reasons for designing an algorithm for the minimization of a linear combination policy attributes rather than only for specific attributes criterion, like the minimize mean and variance, are three-folded. First, the linear combination problem is more realistic, as the expected travel time is usually the primary concern of travelers in a stochastic time-dependent network, while the reliability criteria are secondary. Second, a linear combination is a reasonable way of combining multiple objectives. Third, a linear combination is algorithmically easy to control in the decision making process.

In this chapter, two computationally efficient algorithms are presented for determining the minimum mean-variance paths for all origins to a single destination in networks where the arc weights are discrete random variables whose probability distribution functions vary with time. At termination of the algorithm, efficient solutions (or non-dominated solutions) are generated. Such efficient solutions can be presented to the traveler, who may then make the appropriate choice. For the multiobjective routing problem, an efficient solution provides a route which is such that no other route provides a lower mean travel time and a lower variance

5.2.1 The TAMMV1 Algorithm

The TAMMV1 algorithm, like the PMV algorithm, is a specialized label correcting algorithm. Here, the two label vectors, $[\xi_i(t), \omega_i(t)]_{t\in T}$, are associated with every node. At termination of the algorithm, the "best" next arc is generated for any node to the destination node for time t using the two label vectors, $[\xi_i(t), \omega_i(t)]_{t\in T}$. These arcs are not necessarily associated with a single path, and thus, the paths cannot be reconstructed upon termination.

Let the vector $[e_i(t), v_i(t)]_{t\in T}$ be the temporary label from node i. Denoted by $\xi_i(t), \omega_i(t)$, the current label at the end of the kth iteration. Since we have a clear understanding about the first comparison rule, we present the recursive equation to compute the linear combination of the mean and variance of a routing. From the previous chapter, we know the following recursive equation for the mean and variance of the travel time

$$e_i(t) = \sum_k \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_k \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right], \ \forall i \in N, \ \forall t \in T$$

As we verify the principle of optimality of the minimum expected travel time and the minimum variance in Chapter 4, we can say that the principle of optimality holds for the minimization of the linear combination of the mean and variance routing policy. The linear combination of the above two equations is $e_i(t) + \alpha * \sqrt{v_i(t)}$. The optimality conditions are:

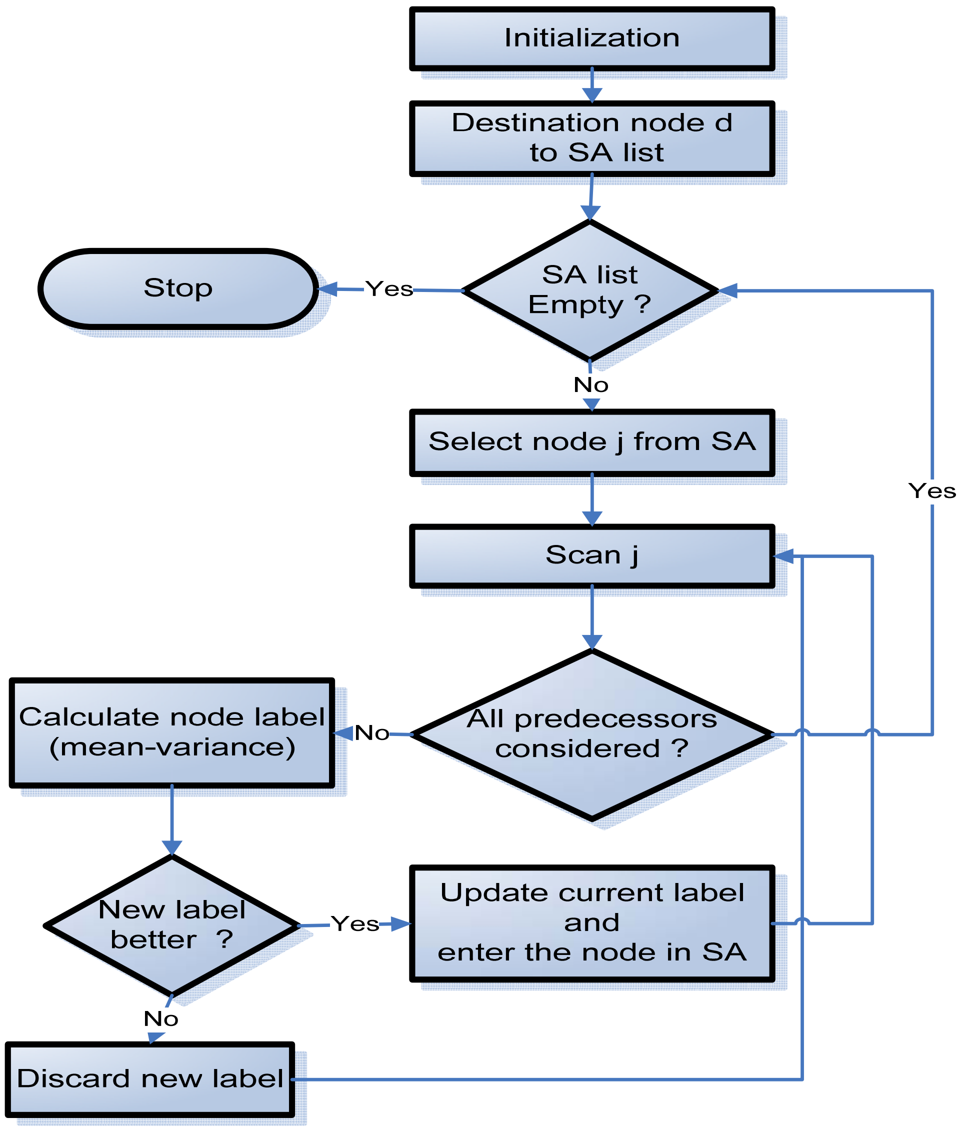
$$e_i(t) = \min_{j \in A(i)} \left(\sum_k \left[\tau_{ij}^k(t) * p_{ij}^k(t)\right] + \sum_k \left[e_j\left(t + \tau_{ij}^k(t)\right) * p_{ij}^k(t)\right] \right), \quad \forall i \in N, \ \forall t \in T$$

$$v_i(t) = \min_{j \in A(i)} \left(\sum_k p_{ij}^k(t) * v_j(t + \tau_{ij}^k(t)) + \sum_k p_{ij}^k(t) * \left(\tau_{ij}(t) + e_j(t + \tau_{ij}^k(t)) - e_i(t)\right)^2 \right), \forall i \in N, \ \forall t \in T$$

$$e_i(t), \ v_i(t) = \min_{j \in A(i)} \left(e_i(t) + \alpha * \sqrt{v_i(t)}\right), \quad \forall i \in N, \ \forall t \in T$$

In this linear combination, the parameter α is selected by a traveler. If a traveler chooses α=0, his selection of the “best” next arc is based on only the expected travel time. However, selecting a large α is for concerning more variance than expected travel time.

A flow chart the basic procedure steps of the TAMMV1 algorithm is presented in Figure 5.3. An example problem in figure 4.2 is shown in section 5.5 to illustrate this procedure.



Initialization
Destination node d to SA list
SA list Empty ?
Stop
Yes
No
Select node j from SA
Yes
Scan j
All predecessors considered ?
No
Calculate node label (mean-variance)
New label better ?
Yes
Update current label and enter the node in SA
No
Discard new label

Figure 5.3. Flow chart of the TAMMV1 algorithm steps

<u>Algorithm TAMMV1</u>

```
begin
        Procedure Initialization
                begin
                        create the NODE_LIST, SA_LIST
                        put all nodes i to NODE_LIST
                        set SA_LIST=Ø
                        set each node i,
                                $\xi_i(t) = \infty, \ \forall i \in N - d, \ t \in T$
                        $\xi_d(t) = 0,\ \forall t \in T$
                        $\omega_i(t) = \infty,\ \forall i \in N - d,\ t \in T$
                        $\omega_d(t) = 0,\ \forall t \in T$
                        $\pi_i(t) = \infty,\ \forall i \in N - d, t \in T$
                        $\pi_d(t) = \phi,\ \forall t \in T$
                        Insert destination node, $d$ to set SA_LIST
                end

        while SA_LIST≠Ø do
        begin
                select the first node of the set SA_LIST
                call this node the current node, $j$
                scan the current node, $j$
                begin
                for each $i$ unlabeled do
                        if succ($i$)=$j$, ($i$,$j$)∈A, then
                        begin
                mark node $i$ labeled
                end

                procedure Update Node Labels
                        For all $i$ labeled
                        begin
                                update the vector $[\xi_i(t),\ \omega_i(t), \pi_i(t)]_{t \in T}$
                                calculate $e_i(t),\ v_i(t)\ \ \forall t \in T$ as follows
                                $e_i(t) = \sum_k \left[\left(\tau_{ij}^k(t) + \left(\xi_j(t + \tau_{ij}^k(t)\right)\right) \cdot \rho_{ij}^k(t)\right]$
                                $v_i(t) = \sum_k \left[\rho_{ij}^k(t) * \omega_j\left(t + \tau_{ij}^k(t)\right)\right]$
                                      $+ \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_j\left(t + \tau_{ij}^k(t)\right) - e_i(t)\right)^2\right]$

                                    where $k$ is the set of indices of possible travel
                                    times on arc($i$,$j$) at time $t$.

                                if $\mu_i(t) < \xi_i(t)$ and $\upsilon_i(t) < \omega_i(t)$ then
                                $\xi_i(t) = \mu_i(t),\ \omega_i(t) = \upsilon_i(t),\ \pi_i(t) = (i, j)$

                                if [ $\mu_i(t) < \xi_i(t)$ and $\upsilon_i(t) > \omega_i(t)$ ] or
                                        [ $\mu_i(t) > \xi_i(t)$ and $\upsilon_i(t) < \omega_i(t)$ ] then
                        compare $\mu_i(t) + \alpha * \sqrt{\nu_i(t)} < \xi_i(t) + \alpha * \sqrt{\omega_i(t)}$ then
                            $\xi_i(t) = \mu_i(t),\ \omega_i(t) = \upsilon_i(t),\ \pi_i(t) = (i, j)$
                        otherwise keep previous information

                        if $i \notin SA\_LIST$ then
                        put $i$ in set SA_LIST
                        end

                Remove j from SA_LIST
                Unlabeled all nodes
            end
        end
```

Proposition 5.1. The TAMMV1 algorithm terminates in a finite number of steps.

Proof. The algorithm terminates in a finite number of steps if the SA_LIST is empty in a finite number of steps. Suppose the SA_LIST is not empty in a finite number of steps, then at least one node must be inserted in the SA_LIST an infinite number of times. This implies that the label at the node has improved by at least a positive real-value of travel time. If the improvement at the node continues an infinite number of times, then the travel time on the path would eventually become negative, which contradicts the assumption of the positive travel times. This contradicts the supposition that the SA_LIST is not empty in a finite number of steps, and hence, shows that the TAMMV1 algorithm terminates in a finite number of steps.

Proposition 5.2 The TAMMV1 algorithm with a basic FIFO SA_LIST structure has the worst-case computational complexity $\mathrm{O}(k \cdot TI^2 \cdot n^3)$, where TI is the number of time intervals into which the peck period is discretized, n is the number of nodes in the network, and k is the maximum number of possible values of the arc travel time random variable for the time interval.

Proof. Once the destination node is removed from the SA_LIST, it will never again be updated. All of its predecessor nodes are added to the SA_LIST for updating. Thus, the SA_LIST contains at most the n-1 nodes. From all the nodes initially inserted in the SA_LIST, the one with the least label for a given departure time will be updated permanently. This is repeated for the remaining n-1 nodes. Since there are at most TI(n-1) labels that can be improved, at most TI$(n-1)^2$ will be inserted in the SA_LIST. The procedure "Update Node Labels" requires a maximum of (TI)(k)(n-1) computations for every node that is scanned because in the worst case, each node can be reached by n-1 nodes and each nod has TI labels, requiring k computations. Thus, the complexity of this algorithm is $\mathrm{O}\big((TI)\cdot(n-1)^2\cdot(TI)\cdot(k)\cdot(n-1)\big)$, or $\sim \mathrm{O}(k\cdot TI^2\cdot n^3)$.

5.2.2 The TAMMV2 Algorithm

The TAMMV2 algorithm can be viewed as an efficient specialized label-setting algorithm for determining the "best" next arc routing hyperpaths from all i to a select destination, d. Similar to the TAMMV1 algorithm, a vector label is associated with each node, the components of which correspond to mean and variance of traversal times from the associated nodes to d, at a given departure time. The iterative structure of this procedure is based on that of the DOT (Decreasing Order of Time) algorithm of Chabini (1997) for determining least-time paths in deterministic, time-varying networks. However, the computation and interpretation of the label values in the TAMMV2 algorithm account for the stochastic nature of the arc traversal times. Unlike the TAMMV1 algorithm, where the components of all vector labels are temporarily set until termination (at which time all vector label components become permanently set), after each iteration of the main loop of the TAMMV2 algorithm, where the labels are updated for a specific departure time, t, the component of each vector label associated with t is permanently set for all i.

Since all possible arc travel times are strictly positive, $e_i(t)$ and $v_i(t)$ may be determined entirely from $e_i(s)$ and $v_i(s)$ for later values of time $s>t$. Therefore, if we first compute $e_i(M)$ for all $i \in N$ as a base case, we can proceed to compute $e_i(M-1)$, $e_i(M-2)$,..., $e_i(0)$, in a decreasing order of time, until a complete solution is found. A flow chart the basic procedure steps of the TAMMV2 algorithm is presented in Figure 5.4.

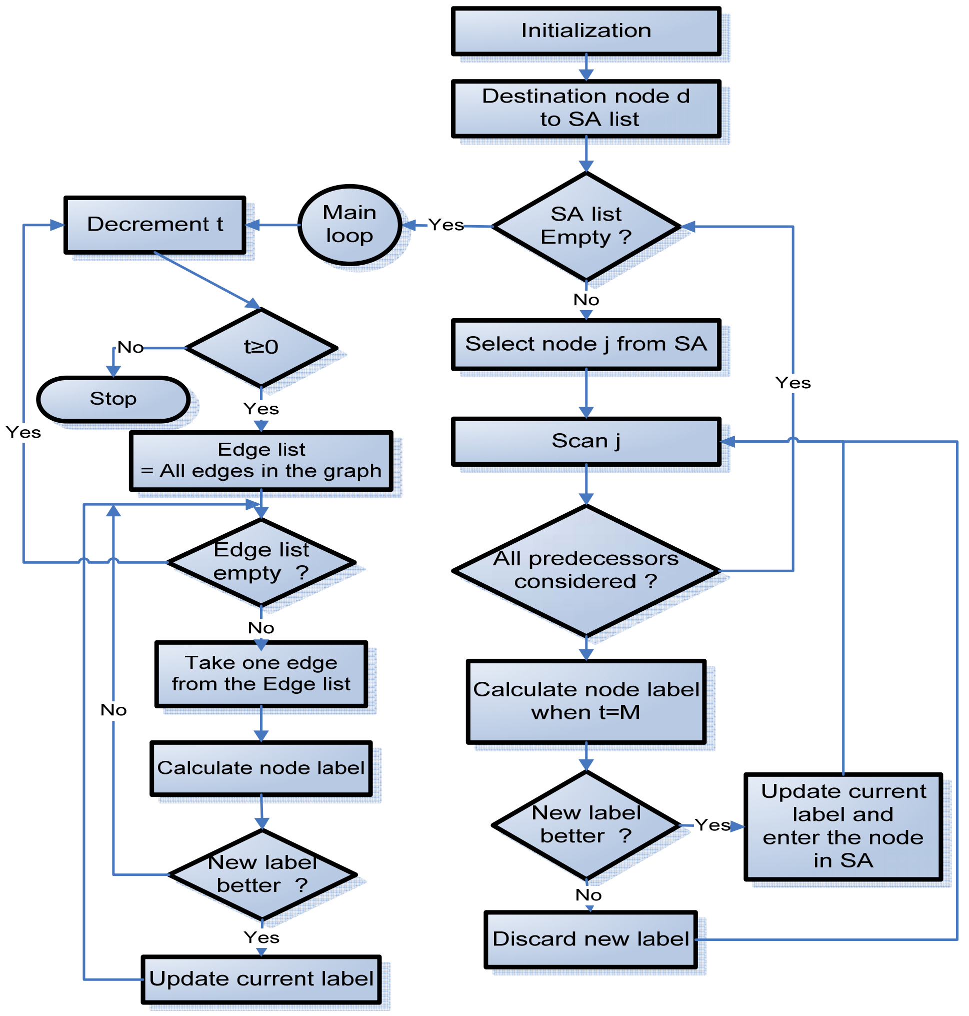


Figure 5.4. Flow chart of the steps of TAMMV2 algorithm

Algorithm TAMMV2

```
begin
    Procedure Initialization
        begin
        create the NODE_LIST, SA_LIST
            put all nodes i to NODE_LIST
            set time space T = {1, 2, …., M}
            set SA_LIST=Ø
            set each node i,
                $\xi_i(t) = \infty,\ \forall i \in N - d, t \in T$
                $\xi_d(t) = 0,\ \forall t \in T$
                $\omega_i(t) = \infty,\ \forall i \in N - d, t \in T$
                $\omega_d(t) = 0,\ \forall t \in T$
                $\pi_i(t) = \infty,\ \forall i \in N, t \in T$
                $\pi_d(t) = \phi,\ \forall t \in T$
            Insert destination node, d to set SA_LIST
        end

        While $SA\_LIST \neq \phi$ do
        begin
        select the first node of the S
            call this node the current node, j
            scan the current node
            begin
            for each i unlabeled do
                    if succ(i)=j, (i,j)∈A, then
                    begin
                        mark node i labeled
                    end
            end
            for all i labeled update the vector $\left[\xi_i(t),\ \omega_i(t), \pi_i(t)\right]_{t=M}$
            begin
            $e_i\left(t = M\right) = \sum_k \left[\left(\tau_{ij}^k(M) + (\xi_j\left(M + \tau_{ij}^p(M)\right)\right) * \rho_{ij}^k(M)\right]$
            $v_i\left(t = M\right) = \sum_k \left[\rho_{ij}^k(M) * v_j\left(M + \tau_{ij}^k(M)\right)\right]$
                $+ \sum_k \left[\rho_{ij}^k(M) * \left(\tau_{ij}^k(M) + \xi_j\left(M + \tau_{ij}^k(M)\right) - e_i(M)\right)^2\right]$
```

```
if $e_i(M) < \xi_i(M)$ and $\nu_i(M) < \omega_i(M)$ then
        $\xi_i(M) = e_i(M),\ \omega_i(M) = \nu_i(M),\ \pi_i(M) = (i, j)$

if [ $e_i(M) < \xi_i(M)$ and $\nu_i(M) > \omega_i(M)$ ] or
        [ $e_i(M) > \xi_i(M)$ and $\nu_i(M) < \omega_i(M)$ ] then

compare $e_i(M) + \alpha * \sqrt{\nu_i(M)} < \xi_i(M) + \alpha * \sqrt{\omega_i(M)}$ then
                $\xi_i(M) = e_i(M),\ \omega_i(M) = \nu_i(M),\ \pi_i(M) = j$
otherwise keep previous information

if $i \notin SA\_LIST$ then
put $i$ in set SA_LIST

end


Procedure Main Loop
    begin
        set each node $i$,
            $\xi_i(t) = \infty,\ \forall i \in N - d,\ t < M$
            $\xi_d(t) = 0,\ \forall t < M$
            $\omega_i(t) = \infty,\ \forall i \in N - d,\ t < M$
            $\omega_d(t) = 0,\ \forall t < M$
            $\pi_i(t) = \infty,\ \forall i \in N - d, t < M$
            $\pi_d(t) = \phi,\ \forall t < M$

        for $t = M$ - 1 to 0
            for all links $(i, j) \in A$
            update the vector $\left[\xi_i(t),\ \omega_i(t), \pi_i(t)\right]_{t \in T}$
            $e_i(t) = \sum_k \left[\left(\tau_{ij}^k(t) + \left(\xi_j(t + \tau_{ij}^k(t))\right)\cdot \rho_{ij}^k(t)\right]$
            $\nu_i(t) = \sum_k \left[\rho_{ij}^k(t) * \omega_j\left(t + \tau_{ij}^k(t)\right)\right]$
                $+ \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_j\left(t + \tau_{ij}^k(t)\right) - e_i(t)\right)^2\right]$
            $\pi_i(t) = Arg\min_{j \in A(i)} \left\{ \begin{array}{l} \sum_k \left[\rho_{ij}^k(t) * \nu_j\left(t + \tau_{ij}^k(t)\right)\right] \\ + \sum_k \left[\rho_{ij}^k(t) * \left(\tau_{ij}^k(t) + \xi_j\left(t + \tau_{ij}^k(t)\right) - e_i(t)\right)^2\right] \end{array} \right\}$
            if $e_i(t) < \xi_i(t)$ and $v_i(t) < \omega_i(t)$ then
            $\xi_i(t) = e_i(t), \omega_i(t) = v_i(t), \pi_i(t) = (i, j)$

            if [ $e_i(t) < \xi_i(t)$ and $v_i(t) > \omega_i(t)$ ] or
            [ $e_i(t) > \xi_i(t)$ and $v_i(t) < \omega_i(t)$ ] then

            compare $e_i(t) + \alpha * \sqrt{v_i(t)} < \xi_i(t) + \alpha * \sqrt{\omega_i(t)}$ then
                    $\xi_i(t) = e_i(t), \omega_i(t) = v_i(t), \pi_i(t) = (i, j)$
            otherwise keep previous information
    end
```

Proposition 5.3 The TAMMV2 algorithm terminates in a finite number of steps.

Proof. The proof is a straightforward extension of that given in proposition 6 for the TAMMV1 algorithm with the difference that only selected $t(= M) \in T$ is considered in the initialization procedure. Therefore, we only need to show the main loop of the algorithm. In the main loop, all node labels are updated at each time interval. Since the number of nodes and travel times are finite, all node labels are updated to the finite number of times with $(n-1)*(TI-1)$. This shows that the TAMMV2 algorithm terminates in a finite number of steps.

Proposition 5.4 The worst-case computational complexity of the TAMMV2 algorithm is $\mathrm{O}(k \cdot n^3 + m \cdot TI \cdot k)$.

Proof. The complexity of Algorithm TAMMV2 is straightforward. At initialization of this algorithm, the running time is $\mathrm{O}(k \cdot n^3)$ because the node label can be updated for only a selected travel time t=M. In the main loop, at each time period of the dynamic period (i.e. *t* < *M*), each arc is visited exactly once with the k mathematic operations. At the end of each such iteration, where labels are updated at a specific departure time, t, and the component of each vector label associated with *t* is permanently set for all nodes. Thus, there are TI-1 iterations of the main loop resulting in $m \cdot (TI - 1) \cdot k$ computations. Therefore the running time of the main loop is $\mathrm{O}(m \cdot TI \cdot k)$. To sum up, the worst-case computational complexity of the TAMMV2 algorithm is $\mathrm{O}(k \cdot n^3 + m \cdot TI \cdot k)$*)*.

An example problem is shown in section 5.3 to illustrate this procedure.

5.3 Conclusions

In this chapter, a method for comparing probability distributions which is less strict than the classical stochastic dominance is presented. This method has an important practical application in determining a set of non-dominated routes in a network when there are uncertain measures. Also, two efficient algorithms for determining the time-adaptive minimum mean-variance hyperpaths were presented. An example was given to show that the adaptive strategies can lead to improved routing decisions over *a priori* path selection. Such a procedure is applicable to many problems that can be represented as stochastic time-dependent networks and is of particular interest in the transportation and data communication systems. An understanding of these two algorithms provides an important step in developing efficient techniques for real-time routing of vehicles in Intelligent Transportation Systems and real-time routing protocols for packets in data networks.

Example for TAMMV1

Initialization

Node_List N = {1,2,3,4}
Arc_List A = {a, b, c, d, e}
Time_space T = {0,1,2,3,4,5,6,7}
Destination_Node D = 4

Scan_Available List SA = ϕ

Set each node i,

$\xi_i(t) = \infty, \ \forall i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\xi_4(t) = 0, \ \forall t \in \{0,1,2,3,4,5,6,7\}$

$\xi_i(t)$ is label of node *i*, at time *t*, where until termination of the algorithm (the expected travel time from node *i* to the destination at time *t*)

$\omega_i(t) = \infty, \ \forall i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\omega_4(t) = 0, \ \forall t \in \{0,1,2,3,4,5,6,7\}$

$\omega_i(t)$ is label of node *i*, at time *t*, where until termination of the algorithm (the variance of travel time from node *i* to the destination at time *t*)

$\pi_i(t) = \infty, \ \forall i \in \{1, 2, 3\}, t \in \{0,1,2,3,4,5,6,7\}$

$\pi_4(t) = \phi, \ \forall t \in \{0,1,2,3,4,5,6,7\}$

$\pi_i(t)$ indicates the arc to be followed from node *i* at time *t*

Create the Scan_Avabable list, SA, and insert the destination node D

SA = {4}

Step 1

If SA list is empty, go to step 3
Otherwise,
Select the first node 4 from the SA list
Call this is current node, *j* = 4

Step 2

For each *i*, $(i, j) \in A$
$i =$ {2, 3}
Determine the lower bound on the expected time and variance to node 4

Update the vector $\xi_i(t)$ and $\omega_i(t)$

Select node 2

Calculate

Expected time :

$$\mu_i(t) = \sum_p \left[\left(\tau_{ij}^p(t) + (\xi_j\left(t + \tau_{ij}^p(t)\right) \right) * \rho_{ij}^p(t) \right]$$

Variance :

$$\nu_i(t) = \sum_p \left[\rho_{ij}^p(t) * \nu_j\left(t + \tau_{ij}^p(t)\right) \right] + \sum_p \left[\rho_{ij}^p(t) * \left(\tau_{ij}^p(t) + \xi_j\left(t + \tau_{ij}^p(t)\right) - \mu_i(t) \right)^2 \right]$$

i = 2

t = 2

$$\begin{aligned}
\mu_2(t=2) &= \left[\left(\tau_{24}^1(2) + \left(\xi_4(2 + \tau_{24}^1(2)) \right) \right) * \rho_{24}^1(2) \right] + \left[\left(\tau_{24}^2(2) + \left(\xi_4(2 + \tau_{24}^2(2)) \right) \right) * \rho_{24}^2(2) \right] \\
&= \left(3 + \xi_4(2+3) \right) * 0.8 + \left(7 + \xi_4(2+7) \right) * 0.2 \\
&= (3+0) * 0.8 + (7+0) * 0.2 \\
&= 3.8
\end{aligned}$$

$$\begin{aligned}
\nu_2(t=2) &= \left[\left(\tau_{24}^1(2) - \mu_2(2) \right)^2 * \rho_{24}^1(2) \right] + \left[\left(\tau_{24}^2(2) - \mu_2(2) \right)^2 * \rho_{24}^2(2) \right] \\
&= \left((3 - 3.8)^2 * 0.8 \right) + \left((7 - 3.8)^2 * 0.2 \right) \\
&= 2.56
\end{aligned}$$

$$\begin{aligned}
\nu_2(t=2) &= \left[\rho_{24}^1(2) * \nu_4\left(2 + \tau_{24}^1(2)\right) \right] + \left[\rho_{24}^2(2) * \nu_4\left(2 + \tau_{24}^2(2)\right) \right] \\
&\quad + \left[\rho_{24}^1(2) * \left(\tau_{24}^1(2) + \mu_4\left(2 + \tau_{24}^1(2)\right) - \mu_2(2) \right)^2 \right] \\
&\quad + \left[\rho_{24}^2(2) * \left(\tau_{24}^2(2) + \mu_4\left(2 + \tau_{24}^2(2)\right) - \mu_2(2) \right)^2 \right] \\
&= \left[0.8 * \nu_4(2+3) \right] + \left[0.2 * \nu_4(2+7) \right] \\
&\quad + \left[0.8 * \left(3 + \mu_4(2+3) - \mu_2(2) \right)^2 + 0.2 * \left(7 + \mu_4(2+7) - \mu_2(2) \right)^2 \right] \\
&= (0.8 * 0) + (0.2 * 0)) + \left(0.8 * (3 + 0 - 3.8)^2 \right) + \left(0.2 * (7 + 0 - 3.8)^2 \right) \\
&= 2.56
\end{aligned}$$

$$\lambda_i(t) = \mu_i(t) + \alpha * \sqrt{\nu_i(t)}$$

Update Label

If $\mu_i(t) < \xi_i(t),\ and\ \nu_i(t) < \omega_i(t)$

Then $\xi_i(t) = \mu_i(t),\ and\ \omega_i(t) = \nu_i(t), and\ \pi_i(t) = (i, j)$

If $\mu_i(t) < \xi_i(t),\ and\ \nu_i(t) > \omega_i(t)$

Compare $\mu_i(t) + \alpha * \sqrt{\nu_i(t)} < \xi_i(t) + \alpha * \sqrt{\omega_i(t)}$

Then $\xi_i(t) = \mu_i(t),\ and\ \omega_i(t) = \nu_i(t), and\ \pi_i(t) = (i, j)$

If $\mu_i(t) > \xi_i(t),\ and\ \nu_i(t) < \omega_i(t)$

Compare $\mu_i(t)+\alpha*\sqrt{\nu_i(t)} < \xi_i(t)+\alpha*\sqrt{\omega_i(t)}$

Then $\xi_i(t)=\mu_i(t),\ and\ \omega_i(t)=\nu_i(t), and\ \pi_i(t)=(i,j)$

Otherwise Keep previous Value

Since $\mu_2(t=2) = 3.8 < \xi_2(t=2)=\infty$,

$\nu_2(t=2) = 2.56 < \omega_i(t=2)=\infty$

Update $\xi_2(2)=3.8$, $\omega_2(2)=2.56$, $\pi_2(2)=d$

t = 3

$$\begin{aligned}\mu_2(t=3) &= \left[\left(\tau_{24}^1(3)+\left(\xi_4(3+\tau_{24}^1(3)\right)\right)*\rho_{24}^1(3)\right]+\left[\left(\tau_{24}^2(3)+\left(\xi_4(3+\tau_{24}^2(3)\right)\right)*\rho_{24}^2(3)\right]\\ &= \left(6+\xi_4(2+6)\right)*0.4+\left(7+\xi_4(2+7)\right)*0.6\\ &= (6+0)*0.4+(7+0)*0.6\\ &= 6.6\end{aligned}$$

$$\begin{aligned}\nu_2(t=3) &= \left[\rho_{24}^1(3)*\nu_4\left(3+\tau_{24}^1(3)\right)\right]+\left[\rho_{24}^2(3)*\nu_4\left(3+\tau_{24}^2(3)\right)\right]\\ &\quad+\left[\rho_{24}^1(3)*\left(\tau_{24}^1(3)+\mu_4\left(3+\tau_{24}^1(3)\right)-\mu_2(3)\right)^2\right]\\ &\quad+\left[\rho_{24}^2(3)*\left(\tau_{24}^2(3)+\mu_4\left(3+\tau_{24}^2(3)\right)-\mu_2(3)\right)^2\right]\\ &= \left[0.4*\nu_4(3+6)\right]+\left[0.6*\nu_4(3+7)\right]\\ &\quad+\left[0.4*\left(6+\mu_4(3+6)-\mu_2(3)\right)^2+0.6*\left(7+\mu_4(3+7)-\mu_2(3)\right)^2\right]\\ &= (0.4*0)+(0.6*0))+\left(0.4*(6+0-6.6)^2\right)+\left(0.6*(7+0-6.6)^2\right)\\ &= 0.24\end{aligned}$$

$\lambda_i(t)=\mu_i(t)+\alpha*\sqrt{\nu_i(t)}$

Update Label

Since $\mu_2(t=3) = 6.6 < \xi_2(t=3)=\infty$,

$\nu_2(t=3) = 0.24 < \omega_i(t=3)=\infty$

Update $\xi_2(3)=6.6$, $\omega_2(3)=0.24$, $\pi_2(3)=d$

If *i* is not in SA list, Put *i* in SA list

SA = {2}

Select node 3

i = 3

t = 4

$$\mu_3(t=4)=\left[\left(\tau_{34}^1(4)+\left(\xi_4(4+\tau_{34}^1(4)\right)\right)*\rho_{34}^1(4)\right]+\left[\left(\tau_{34}^2(4)+\left(\xi_4(4+\tau_{34}^2(4)\right)\right)*\rho_{34}^2(4)\right]$$
$$=\left(4+\xi_4(2+4)\right)*0.2+\left(6+\xi_4(2+6)\right)*0.8$$
$$=(4+0)*0.2+(6+0)*0.8$$
$$=5.6$$
$$\nu_3(t=4)=\left[\rho_{34}^1(4)*\nu_4\left(4+\tau_{34}^1(4)\right)\right]+\left[\rho_{34}^2(4)*\nu_4\left(4+\tau_{34}^2(4)\right)\right]$$
$$+\left[\rho_{34}^1(4)*\left(\tau_{34}^1(4)+\mu_4\left(4+\tau_{34}^1(4)\right)-\mu_3(4)\right)^2\right]$$
$$+\left[\rho_{34}^2(4)*\left(\tau_{34}^2(4)+\mu_4\left(6+\tau_{34}^2(4)\right)-\mu_3(4)\right)^2\right]$$
$$=\left[0.2*\nu_4(4+4)\right]+\left[0.8*\nu_4(4+6)\right]$$
$$+\left[0.2*\left(4+\mu_4(4+4)-\mu_3(4)\right)^2+0.8*\left(6+\mu_4(4+6)-\mu_3(4)\right)^2\right]$$
$$=\left(0.2*0\right)+\left(0.8*0\right)\right)+\left(0.2*(4+0-5.6)^2\right)+\left(0.8*(6+0-5.6)^2\right)$$
$$=0.64$$
$$\lambda_i(t)=\mu_i(t)+\alpha*\sqrt{\nu_i(t)}$$

Update Label

Since $\mu_3(t=4)=5.6<\xi_3(t=4)=\infty$,
$\nu_3(t=4)=0.64<\omega_3(t=4)=\infty$
Update $\xi_3(4)=5.6$, $\omega_3(4)=0.64$, $\pi_2(2)=e$

t = 5

$$\mu_3(t=5)=\left(5+\xi_4(5+5)\right)*0.3+\left(8+\xi_4(5+8)\right)*0.7$$
$$=(5+0)*0.3+(8+0)*0.7$$
$$=7.1$$

$$\nu_3(t=5)=\left[0.3*\nu_4(5+5)\right]+\left[0.7*\nu_4(5+8)\right]$$
$$+\left[0.3*\left(5+\mu_4(5+5)-\mu_3(5)\right)^2+0.7*\left(8+\mu_4(5+8)-\mu_3(5)\right)^2\right]$$
$$=\left(0.3*(5+0-7.1)^2\right)+\left(0.7*(8+0-7.1)^2\right)$$
$$=1.89$$

Update Label
Since $\mu_3(t=5)=7.1<\xi_3(t=5)=\infty$,
$\nu_3(t=5)=1.89<\omega_3(t=5)=\infty$
Update $\xi_3(5)=7.1$, $\omega_3(5)=1.89$, $\pi_3(5)=e$

t = 6

$$\mu_3(t=6) = (1+\xi_4(6+1))*0.9 + (2+\xi_4(6+2))*0.1$$
$$= (1+0)*0.9 + (2+0)*0.1$$
$$= 1.1$$

$$\nu_3(t=6) = [0.9*\nu_4(6+1)] + [0.1*\nu_4(6+2)]$$
$$+ \left[0.9*(1+\mu_4(6+1)-\mu_3(6))^2 + 0.1*(2+\mu_4(6+2)-\mu_3(6))^2\right]$$
$$= (0.9*(1+0-1.1)^2) + (0.1*(2+0-1.1)^2)$$
$$= 0.09$$

Update Label

Since $\mu_3(t=6)$ =1.1 < $\xi_3(t=6) = \infty$

$\nu_3(t=6)$ =0.09 < $\omega_3(t=6) = \infty$

Update $\xi_3(6)$ =1.1, $\omega_3(6)$ =0.09, $\pi_3(6) = e$

t = 7

$$\mu_3(t=7) = (3+\xi_4(7+3))*0.3 + (4+\xi_4(7+4))*0.7$$
$$= (3+0)*0.3 + (4+0)*0.7$$
$$= 3.7$$
$$\nu_3(t=7) = [0.3*\nu_4(7+3)] + [0.7*\nu_4(7+4)]$$
$$+ \left[0.3*(3+\mu_4(7+3)-\mu_3(7))^2 + 0.7*(4+\mu_4(7+4)-\mu_3(7))^2\right]$$
$$= (0.3*(3+0-3.7)^2) + (0.7*(4+0-3.7)^2)$$
$$= 0.21$$

Update Label

Since $\mu_3(t=7)$ =3.7 < $\xi_3(t=7) = \infty$

$\nu_3(t=7)$ =0.21 < $\omega_3(t=7) = \infty$

Update $\xi_3(7)$ =3.7, $\omega_3(7)$ =0.21, $\pi_3(7) = e$

If *i* is not in SA list, Put node 3 in SA list

SA = {2, 3}

GO TO STEP 1

Select the first node 2 from the SA list

Call this is current node, ***j* = 2, SA ={3}**

<u>Step 2</u>

For each i, $(i, j) \in A$

$i =$ $\{1\}$

Determine the lower bound on the expected time and variance

Update the vector $\xi_i(t)$ and $\omega_i(t)$

Select node 1

i = 1
t = 0

$$\begin{aligned}
\mu_1(t=0) &= \left[\left(\tau_{12}^1(0) + \left(\xi_2(0+\tau_{12}^1(0)\right)\right) * \rho_{12}^1(0)\right] + \left[\left(\tau_{12}^2(0) + \left(\xi_2(0+\tau_{12}^2(0)\right)\right) * \rho_{12}^2(0)\right] \\
&= \left(2+\xi_2(0+2)\right)*0.5 + \left(3+\xi_2(0+3)\right)*0.5 \\
&= (3+3.8)*0.5 + (7+6.6)*0.5 \\
&= 7.7
\end{aligned}$$

$$\begin{aligned}
\nu_1(t=0) &= \left[\rho_{12}^1(0) * \nu_2\left(0+\tau_{12}^1(0)\right)\right] + \left[\rho_{12}^2(0) * \nu_2\left(0+\tau_{12}^2(0)\right)\right] \\
&\quad + \left[\rho_{12}^1(0) * \left(\tau_{12}^1(0) + \mu_2\left(2+\tau_{12}^1(0)\right) - \mu_1(0)\right)^2\right] \\
&\quad + \left[\rho_{12}^2(0) * \left(\tau_{12}^2(0) + \mu_2\left(3+\tau_{12}^2(0)\right) - \mu_1(0)\right)^2\right] \\
&= \left[0.5*\nu_2(0+2)\right] + \left[0.5*\nu_2(0+3)\right] \\
&\quad + \left[0.5*(2+\mu_2(0+2)-\mu_1(0))^2 + 0.5*(3+\mu_2(0+3)-\mu_1(0))^2\right] \\
&= (0.5*2.56) + (0.5*0.24)) + \left(0.5*(2+3.8-7.7)^2\right) + \left(0.5*(3+6.6-7.7)^2\right) \\
&= 1.28 + 0.12 + 1.805 + 1.805 \\
&= 5.01
\end{aligned}$$

Update Label

Since $\mu_1(t=0)$ =7.7 $<$ $\xi_1(t=0) = \infty$

$\nu_1(t=0)$ =5.01 $<$ $\omega_1(t=0) = \infty$

Update $\xi_1(0)$ =7.7, $\omega_1(0)$ =5.01, $\pi_1(0) = a$

Node 1 is not in SA list, Put *node 1* in SA list

SA = {3,1}

GO TO STEP 1

Select the first node 3 from the SA list

Call this is current node, ***j* = 3, SA ={1}**

<u>Step 2</u>

For each i, $(i, j) \in A$

$i = \{1, 2\}$

Determine the lower bound on the expected time and variance

Update the vector $\xi_i(t)$ and $\omega_i(t)$

Select node 1

i = 1

t = 0

$$\mu_1(t=0) = \left[\left(\tau^1_{13}(0) + \left(\xi_3(0+\tau^1_{13}(0)\right)\right) * \rho^1_{13}(0)\right] + \left[\left(\tau^2_{13}(0) + \left(\xi_3(0+\tau^2_{13}(0)\right)\right) * \rho^2_{13}(0)\right]$$

$$= \left(5 + \xi_3(0+5)\right) * 0.4 + \left(7 + \xi_3(0+7)\right) * 0.6$$

$$= (5 + 7.1) * 0.4 + (7 + 3.7) * 0.6$$

$$= 11.26$$

$$\nu_1(t=0) = \left[\rho^1_{13}(0) * \nu_3\left(0 + \tau^1_{13}(0)\right)\right] + \left[\rho^2_{13}(0) * \nu_3\left(0 + \tau^2_{13}(0)\right)\right]$$

$$+ \left[\rho^1_{13}(0) * \left(\tau^1_{13}(0) + \mu_3\left(5 + \tau^1_{13}(0)\right) - \mu_1(0)\right)^2\right]$$

$$+ \left[\rho^2_{13}(0) * \left(\tau^2_{13}(0) + \mu_3\left(7 + \tau^2_{13}(0)\right) - \mu_1(0)\right)^2\right]$$

$$= \left[0.4 * \nu_3(0+5)\right] + \left[0.6 * \nu_3(0+7)\right]$$

$$+ \left[0.4 * \left(5 + \mu_3(0+5) - \mu_1(0)\right)^2 + 0.6 * \left(7 + \mu_3(0+7) - \mu_1(0)\right)^2\right]$$

$$= (0.4 * 1.89) + (0.6 * 0.21)) + \left(0.4 * (5 + 7.1 - 11.26)^2\right) + \left(0.6 * (7 + 3.7 - 11.26)^2\right)$$

$$= 0.882 + 0.4704$$

$$= 1.3524$$

Update Label

Since $\mu_1(t=0) = 11.26 \quad > \quad \xi_1(t=0) = 7.7$

$\nu_1(t=0) = 1.35 \quad < \quad \omega_1(t=0) = 5.01$

Let $\alpha = 1$, Compare

$$\lambda_i(t) = \mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 11.26 + 1 * \sqrt{1.35} = 12.42$$

$$\mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 12.42 > \xi_i(t) + \alpha * \sqrt{\omega_i(t)} = 7.7 + 1 * \sqrt{5.01} = 9.94$$

So, Keep $\xi_1(0) = 7.7$, $\omega_1(0) = 5.01$, $\pi_1(0) = b$

Node 1 is already in SA list,

SA = {3}

Select node 2

i = 2 (j = 3)

<u>t = 2</u>

$$\mu_2(t=2) = \left[\left(\tau_{23}^1(2) + \left(\xi_3(2+\tau_{23}^1(2)\right)\right) * \rho_{23}^1(2)\right] + \left[\left(\tau_{23}^2(2) + \left(\xi_3(2+\tau_{23}^2(2)\right)\right) * \rho_{23}^2(2)\right]$$
$$= \left(4 + \xi_3(2+4)\right) * 0.8 + \left(5 + \xi_3(2+5)\right) * 0.2$$
$$= (4+1.1) * 0.8 + (5+3.7) * 0.2$$
$$= 5.82$$

$$\nu_2(t=2) = 2.188$$

Update Label

Since $\mu_2(t=2)$ =5.82 $>$ $\xi_2(t=2) = 3.8$

$\nu_2(t=2)$ =2.19 $<$ $\omega_2(t=2) = 2.56$

Let $\alpha = 1$, Compare

$$\lambda_i(t) = \mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 5.82 + 1 * \sqrt{2.19} = 7.30$$

$$\mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 7.30 > \xi_i(t) + \alpha * \sqrt{\omega_i(t)} = 3.8 + 1 * \sqrt{2.56} = 5.4$$

So, Keep $\xi_2(2)$ =3.8, $\omega_2(2)$ =2.56, $\pi_2(2) = d$

<u>t = 3</u>

$$\mu_2(t=3) = \left[\left(\tau_{23}^1(3) + \left(\xi_3(3+\tau_{23}^1(3)\right)\right) * \rho_{23}^1(3)\right] + \left[\left(\tau_{23}^2(3) + \left(\xi_3(3+\tau_{23}^2(3)\right)\right) * \rho_{23}^2(3)\right]$$
$$= \left(1 + \xi_3(3+1)\right) * 0.3 + \left(3 + \xi_3(3+3)\right) * 0.7$$
$$= (1+5.6) * 0.3 + (3+1.1) * 0.7$$
$$= 4.85$$
$$\nu_2(t=3) = 1.57$$

Update Label

Since $\mu_2(t=3)$ =4.85 $<$ $\xi_2(t=3) = 6.6$

$\nu_2(t=3)$ =1.57 $>$ $\omega_2(t=3) = 0.24$

Let $\alpha = 1$, Compare

$$\lambda_i(t) = \mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 4.85 + 1 * \sqrt{1.57} = 6.08$$

$$\mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 6.08 < \xi_i(t) + \alpha * \sqrt{\omega_i(t)} = 6.6 + 1 * \sqrt{0.24} = 7.09$$

So, Updata $\xi_2(3)$ =4.85, $\omega_2(3)$ =1.57, $\pi_2(3) = c$

If *i* is not in SA list, Put node 2 in SA list

SA = {1, 2}

GO TO STEP 1

Select the first node 1 from the SA list
Call this is current node, ***j* = 1, SA ={2}**

Step 2

For each i, $(i, j) \in A$
$i = \phi$

GO TO STEP 1

Select the first node 2 from the SA list
Call this is current node, ***j* = 2, SA ={ }**
Determine the lower bound on the expected time and variance
Update the vector $\xi_i(t)$ and $\omega_i(t)$

Select node 1

i = 1
t = 0

$$\begin{aligned}
\mu_1(t=0) &= \left[\left(\tau_{12}^1(0) + \left(\xi_2(0+\tau_{12}^1(0))\right) * \rho_{12}^1(0)\right] + \left[\left(\tau_{12}^2(0) + \left(\xi_2(0+\tau_{12}^2(0))\right) * \rho_{12}^2(0)\right] \\
&= \left(2+\xi_2(0+2)\right)*0.5 + \left(3+\xi_2(0+3)\right)*0.5 \\
&= (2+3.8)*0.5 + (3+4.85)*0.5 \\
&= 6.825
\end{aligned}$$

$$\begin{aligned}
\nu_1(t=0) &= \left[\rho_{12}^1(0) * \nu_2\left(0+\tau_{12}^1(0)\right)\right] + \left[\rho_{12}^2(0) * \nu_2\left(0+\tau_{12}^2(0)\right)\right] \\
&\quad + \left[\rho_{12}^1(0) * \left(\tau_{12}^1(0) + \mu_2\left(2+\tau_{12}^1(0)\right) - \mu_1(0)\right)^2\right] \\
&\quad + \left[\rho_{12}^2(0) * \left(\tau_{12}^2(0) + \mu_2\left(3+\tau_{12}^2(0)\right) - \mu_1(0)\right)^2\right] \\
&= \left[0.5*\nu_2(0+2)\right] + \left[0.5*\nu_2(0+3)\right] \\
&\quad + \left[0.5*(2+\mu_2(0+2) - \mu_1(0))^2 + 0.5*(3+\mu_2(0+3) - \mu_1(0))^2\right] \\
&= (0.5*2.56) + (0.5*1.57)) + \left(0.5*(2+3.8-6.825)^2\right) + \left(0.5*(3+4.85-6.825)^2\right) \\
&= 1.28 + 0.785 + 0.5253 + 0.5253 \\
&= 3.11
\end{aligned}$$

Update Label
Since $\mu_1(t=0)$ =6.825 < $\xi_1(t=0) = 7.7$
$\nu_1(t=0)$ =3.11 < $\omega_1(t=0) = 5.01$
Update $\xi_1(0)$ =6.825, $\omega_1(0)$ =3.11, $\pi_1(0) = a$

SA = { }

STOP

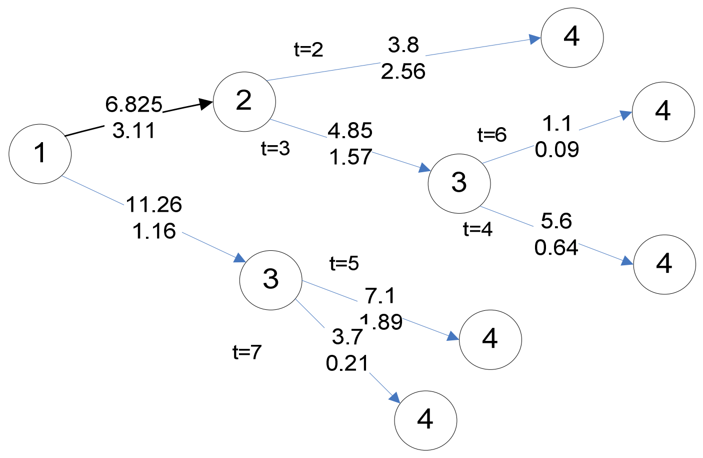


Figure 5.5. Resulting hyperpaths for departure time t=0 from node 1

<u>Example for TAMMV2</u>

Consider the example network in figure 4.2.

TI (t)	arc (1,2)		arc (1,3)		arc (2,3)		arc (2,4)		arc (2,5)	
	T time	Prob.	T time	Prob.	T time	Prob.	T time	Prob.	T time	Prob.
0	4	0.31	7	0.26	1	0.86	3	0.21	8	0.65
	5	0.69	2	0.74	7	0.14	4	0.79	1	0.35
1	1	0.14	7	0.34	4	0.60	2	0.48	4	0.40
	6	0.86	8	0.66	2	0.40	4	0.52	6	0.60
2	4	0.33	4	0.60	3	0.66	8	0.05	2	0.11
	8	0.67	6	0.40	7	0.34	7	0.95	1	0.89
3	2	0.20	7	0.27	5	0.39	9	0.55	7	0.88
	8	0.80	8	0.73	8	0.61	5	0.45	3	0.12
4	3	0.43	1	0.70	6	0.90	3	0.69	6	0.62
	4	0.57	8	0.30	7	0.10	1	0.31	8	0.38
5	1	0.17	9	0.41	6	0.64	7	0.98	2	0.85
	5	0.83	5	0.59	4	0.36	4	0.02	5	0.15

Initialization

Node_List N = {1,2,3,4}
Arc_List A = {a, b, c, d, e}
Time_space T = {0,1,2,3,4,5} , If t > 6, t=5
Destination_Node D = 4
Scan_Available List SA = ϕ

Set each node i,

$\xi_i\left(t=M-1=5\right)=\infty,\ \ \forall i\in\{1,\ 2,\ 3\}$

$\xi_4\left(t=M-1=5\right)=0$

$\xi_i\ (t)$ is label of node *i*, at time *t*, where until termination of the algorithm (the expected travel time from node *i* to the destination at time *t*)

$\omega_i\left(t=M-1=5\right)=\infty,\ \ \forall i\in\{1,\ 2,\ 3\}$

$\omega_4\left(t=M-1=5\right)=0$

$\omega_i\ (t)$ is label of node *i*, at time *t*, where until termination of the algorithm (the variance of travel time from node *i* to the destination at time *t*)

$\pi_i\left(t=M-1=5\right)=\infty,\ \ \forall\ i\in\{1,\ 2,\ 3\}$

$\pi_4\left(t=M-1=5\right)=\phi$

$\pi_i\ (t)$ indicates the arc to be followed from node *i* at time *t*

Set

$e_4\left(t=M-1=5\right)=0\ \ ,\ \ \ e_4\left(t>5\right)=e_4\left(t=5\right),\ \ e_i\left(t>5\right)=e_i\left(t=5\right)\ \ \forall i\in N$

$\gamma_4\left(t=M-1=5\right)=0\ \ \ \ \gamma_4\left(t>5\right)=\gamma_4\left(t=5\right),\ \ \gamma_i\left(t>5\right)=\gamma_i\left(t=5\right)\ \ \forall i\in N$

For all (i, j), $(i,j)\in A$

Calculate

Expected travel time when t=M-1=5:

$$\mu_{ij}\left(t=5\right)=\sum_p\left[\left(\tau_{ij}^p(5)\right)*\rho_{ij}^p(5)\right],\quad \forall(i,j)\in A$$

Variance when t=M-1=5:

$$\nu_{ij}\left(t\right)=\sum_p\left[\rho_{ij}^p(5)*\left(\tau_{ij}^p(5)-\mu_{ij}(5)\right)^2\right],\ \forall(i,j)\in A$$

$\mu_{12}\left(t=5\right)=\sum_p\left[\left(\tau_{12}^p(5)\right)*\rho_{12}^p(5)\right]$=(1*0.17)+(5*0.83) = 4.32

$\mu_{13}\left(t=5\right)=\sum_p\left[\left(\tau_{13}^p(5)\right)*\rho_{13}^p(5)\right]$=(9*0.41)+(5*0.59) = 6.64

$\mu_{23}\left(t=5\right)=\sum_p\left[\left(\tau_{23}^p(5)\right)*\rho_{23}^p(5)\right]$=(6*0.64)+(4*0.36) = 5.28

$\mu_{24}\left(t=5\right)=\sum_p\left[\left(\tau_{24}^p(5)\right)*\rho_{24}^p(5)\right]$=(7*0.98)+(4*0.02) = 6.94

$$\mu_{34}(t=5)=\sum_p\left[\left(\tau_{34}^p(5)\right)*\rho_{34}^p(5)\right]=(2*0.85)+(5*0.15)=2.45$$

$$\nu_{12}(t)=\sum_p\left[\rho_{12}^p(5)*\left(\tau_{12}^p(5)-\mu_{12}(5)\right)^2\right]=(1-4.32)^2*0.17+(5-4.32)^2*0.83=2.26$$

$$\nu_{13}(t)=\sum_p\left[\rho_{13}^p(5)*\left(\tau_{13}^p(5)-\mu_{13}(5)\right)^2\right]=(9-6.64)^2*0.417+(5-6.64)^2*0.59=3.87$$

$$\nu_{23}(t)=\sum_p\left[\rho_{23}^p(5)*\left(\tau_{23}^p(5)-\mu_{23}(5)\right)^2\right]=(6-5.28)^2*0.64+(6-5.28)^2*0.36=0.92$$

$$\nu_{24}(t)=\sum_p\left[\rho_{24}^p(5)*\left(\tau_{24}^p(5)-\mu_{24}(5)\right)^2\right]=(7-6.94)^2*0.98+(4-6.94)^2*0.02=0.18$$

$$\nu_{34}(t)=\sum_p\left[\rho_{34}^p(5)*\left(\tau_{34}^p(5)-\mu_{34}(5)\right)^2\right]=(2-2.45)^2*0.85+(5-2.45)^2*0.15=1.15$$

Create the Scan_Avabable list, SA, and insert the destination node *d*

SA = {4}

<u>Step 1</u>

If SA list is empty, go to step 3
Otherwise,
    Select the first node 4 from the SA list
    Call this is current node, *j* = 4

<u>Step 2</u>

For each *i*, $(i, j) \in A$
$i =$ {2, 3}

Determine the lower bound on the expected time and variance to node 4
Update the vector $\xi_i\ (t = M - 1 = 5)$ and $\omega_i\ (t = M - 1 = 5)$

Select node 2, i =2
Calculate

Expected time :

$$e_i(t)=\sum_p\left[\left(\tau_{ij}^p(t)+(\xi_j\left(t+\tau_{ij}^p(t)\right)\right)*\rho_{ij}^p(t)\right]$$

$$e_i(M-1)=\min_{j\in A(i)}\left\{\mu_{ij}(M-1)+\xi_j(M-1)\right\}$$

$$e_2(5)=\min_{j\in A(i)}\left\{\mu_{24}(5)+\xi_4(5)\right\}=6.94+0=6.94$$

Variance :

$$\gamma_2(5)=\sum_k\left[\rho_{24}^k(5)*\omega_4\left(5+\tau_{24}^k(5)\right)\right]$$
$$+\sum_k\left[\rho_{24}^k(5)*\left(\tau_{24}^k(5)+e_4\left(5+\tau_{24}^k(5)\right)-e_2(5)\right)^2\right]$$
$$=0.98*\omega_4(5+7)+0.02*\omega_4(5+4)$$
$$+0.98*\left(7+\xi_4(5+7)-\xi_2(5)\right)^2+0.02*\left(4+\xi_4(5+4)-\xi_2(5)\right)^2$$
$$= (0.98*0) + (0.02*0)+$$
$$(0.98*(7+0-6.94)^2+(0.02*(4+0-6.94)^2$$
$$= 0.18$$

Update Label

If $e_i(t)<\xi_i(t),\ and\ \gamma_i(t)<\omega_i(t)$

Then $\xi_i(t)=\mu_i(t),\ and\ \omega_i(t)=\nu_i(t), and\ \pi_i(t)=(i,j)$

If $e_i(t)<\xi_i(t),\ and\ \gamma_i(t)>\omega_i(t)$

Compare $e_i(t)+\alpha*\sqrt{\gamma_i(t)} < \xi_i(t)+\alpha*\sqrt{\omega_i(t)}$

Then $\xi_i(t)=\mu_i(t),\ and\ \omega_i(t)=\nu_i(t), and\ \pi_i(t)=(i,j)$

If $e_i(t)>\xi_i(t),\ and\ \gamma_i(t)<\omega_i(t)$

Compare $e_i(t)+\alpha*\sqrt{\gamma_i(t)} < \xi_i(t)+\alpha*\sqrt{\omega_i(t)}$

Then $\xi_i(t)=\mu_i(t),\ and\ \omega_i(t)=\nu_i(t), and\ \pi_i(t)=(i,j)$

Otherwise Keep previous Value

Since $e_2(t=5)=6.94<\xi_2(t=5)=\infty$,

$\gamma_2(t=5)=0.18<\omega_i(t=5)=\infty$

Update $\xi_2(5)=6.94$, $\omega_2(5)=0.18$, $\pi_2(5)=d$

If *i* is not in SA list, Put *i* in SA list

SA = {2}

Select node 3

i = 3

$$e_3(5)=\min_{j\in A(i)}\{\mu_{34}(5)+\xi_4(5)\}=2.45 + 0 = 2.45$$

$$\gamma_3(5)=\sum_k\left[\rho_{34}^k(5)*\omega_4\left(5+\tau_{34}^k(5)\right)\right]$$
$$+\sum_k\left[\rho_{34}^k(5)*\left(\tau_{34}^k(5)+e_4\left(5+\tau_{34}^k(5)\right)-e_3(5)\right)^2\right]$$
$$= (0.85*0)+(0.15*0)+(0.85*(2-2.45)^2+0.15*(5-2.45)^2)$$
$$= 1.15$$

Since $e_3(t=5) = 2.45 < \xi_3(t=5) = \infty$,

$\gamma_3(t=5) = 1.15 < \omega_3(t=5) = \infty$

Update $\xi_3(5) = 2.45$, $\omega_3(5) = 1.15$, $\pi_3(5) = e$

If *i* is not in SA list, Put node 3 in SA list

SA = {2, 3}

GO TO STEP 1

Select the first node 2 from the SA list

Call this is current node, ***j* = 2, SA ={3}**

<u>Step 2</u>

For each *i*, $(i, j) \in A$

Select node 1 , $i = 1$

$$e_1(5) = \min_{j \in A(i)} \{\mu_{12}(5) + \xi_2(5)\} = 4.32 + 6.94 = 11.26$$

$$\gamma_1(5) = \sum_k \left[\rho_{12}^k(5) * \omega_2\left(5 + \tau_{12}^k(5)\right)\right]$$

$$+ \sum_k \left[\rho_{12}^k(5) * \left(\tau_{12}^k(5) + e_2\left(5 + \tau_{12}^k(5)\right) - e_1(5)\right)^2\right]$$

$= (0.17*0.18)+(0.83*0.18)$

$+(0.17*(1+6.94-11.26)^2+0.83*(5+6.94-11.26)^2)$

$= 2.44$

Update node label

$e_1(5) = 11.26 < \xi_1(t=5) = \infty$,

$\gamma_3(t=5) = 2.44 < \omega_3(t=5) = \infty$

Update $\xi_1(5) = 11.26$, $\omega_1(5) = 2.44$, $\pi_1(5) = a$

Node 1 is not in SA list, Put *node 1* in SA list

SA = {3,1}

GO TO STEP 1

Select the first node 3 from the SA list

Call this is current node, ***j* = 3, SA ={1}**

Step 2

For each i, $(i, j) \in A$
$i = \{1, 2\}$

Select node 1 , $i = 1$

$$e_1(5) = \min_{j \in A(i)} \{\mu_{13}(5) + \xi_3(5)\} = 6.64 + 2.45 = 9.09$$

$$\gamma_1(5) = \sum_k \left[\rho_{13}^k(5) * \omega_3\left(5 + \tau_{13}^k(5)\right)\right]$$
$$+ \sum_k \left[\rho_{13}^k(5) * \left(\tau_{13}^k(5) + e_3\left(5 + \tau_{13}^k(5)\right) - e_1(5)\right)^2\right]$$
$$= (0.41*1.15)+(0.59*1.15)$$
$$+(0.41*(9+2.45-9.09)^2+0.59*(5+2.45-9.09)^2)$$
$$= 5.02$$

Update node label

$e_1(5) = 9.09 < \xi_1(t=5) = 11.26,$

$\gamma_1(t=5) = 5.02 > \omega_1(t=5) = 2.44$

Let $\alpha = 1$, Compare

$$\lambda_i(t) = e_i(t) + \alpha * \sqrt{\gamma_i(t)} = 9.09 + 1 * \sqrt{5.02} = 11.33$$
$$\mu_i(t) + \alpha * \sqrt{v_i(t)} = 11.33 < \xi_i(t) + \alpha * \sqrt{\omega_i(t)} = 11.26 + 1 * \sqrt{2.44} = 12.82$$

So, Update $\xi_1(5)$ **=9.09,** $\omega_1(5)$ **=5.02,** $\pi_1(5) =$ ***b***

Node 1 is already in SA list,
SA = {1}

Select node 2 , i = 2 (j = 3)

$$e_2(5) = \min_{j \in A(i)} \{\mu_{23}(5) + \xi_3(5)\} = 5.28 + 2.45 = 7.73$$

Variance :

$$\gamma_2(5) = \sum_k \left[\rho_{23}^k(5) * \omega_3\left(5 + \tau_{23}^k(5)\right)\right]$$
$$+ \sum_k \left[\rho_{23}^k(5) * \left(\tau_{23}^k(5) + e_3\left(5 + \tau_{23}^k(5)\right) - e_2(5)\right)^2\right]$$
$$= 0.64 * \omega_3(5+6) + 0.36 * \omega_3(5+4)$$
$$+ 0.64 * \left(6 + \xi_3(5+6) - \xi_2(5)\right)^2 + 0.36 * \left(4 + \xi_3(5+4) - \xi_2(5)\right)^2$$

= (0.64*1.15) + (0.36*1.15)+
(0.64*(6+2.45-7.73)^2+(0.36*(4+2.45-7.73)^2
= 2.07

Update node label

$e_2(5) = 7.73 > \xi_2(t=5) = 6.94,$

$\gamma_2(t=5) = 2.07 > \omega_2(t=5) = 0.18$

So, keep the $\xi_2(5)$ **=6.94,** $\omega_2(5)$ **=0.18,** $\pi_2(5) =$ ***d***

If *i* is not in SA list, Put node 2 in SA list
SA = {1, 2}

GO TO STEP 1

Select the first node 1 from the SA list
Call this is current node, ***j* = 1, SA ={2}**

<u>Step 2</u>

For each i, $(i, j) \in A$
$i = \phi$

GO TO STEP 1

Select the first node 2 from the SA list
Call this is current node, ***j* = 2, SA ={ }**

Select node 1, i = 1

$$e_1(5) = \min_{j \in A(i)} \{\mu_{12}(5) + \xi_2(5)\} = 4.32 + 6.94 = 11.26$$

$$\gamma_1(5) = \sum_k \left[\rho_{12}^k(5) * \omega_2\left(5 + \tau_{12}^k(5)\right)\right] + \sum_k \left[\rho_{12}^k(5) * \left(\tau_{12}^k(5) + e_2\left(5 + \tau_{12}^k(5)\right) - e_1(5)\right)^2\right]$$

= (0.17*0.18)+(0.83*0.18)
+(0.17*(1+6.94-11.26)^2+0.83*(5+6.94-11.26)^2)
= 2.44

Compare

$$\lambda_i(t) = e_i(t) + \alpha * \sqrt{\gamma_i(t)} = 11.26 + 1 * \sqrt{2.44} = 12.82$$

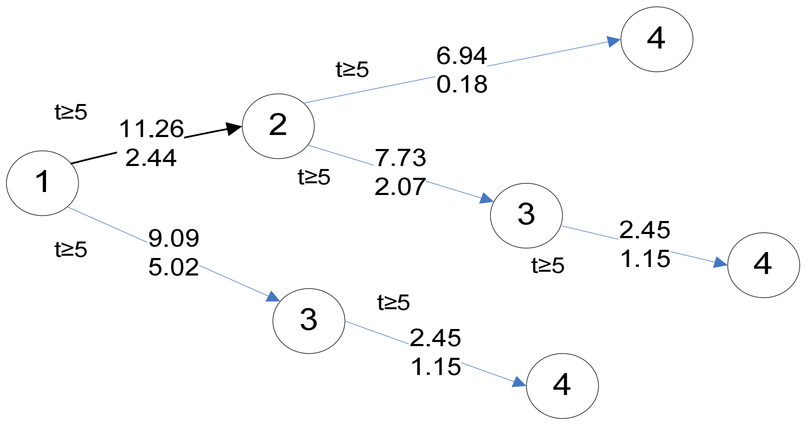
$$\mu_i(t) + \alpha * \sqrt{\nu_i(t)} = 12.82 < \xi_i(t) + \alpha * \sqrt{\omega_i(t)} = 9.09 + 1 * \sqrt{5.02} = 11.33$$

So, Keep $\xi_1(5)$ **=9.09,** $\omega_1(5)$ **=5.02,** $\pi_1(5)$ **=** ***b***

SA = { }

End Initialization

$\xi_1(5)$ **=9.09,** $\omega_1(5)$ **=5.02,** $\pi_1(5)$ **=** ***b***
$\xi_2(5)$ **=6.94,** $\omega_2(5)$ **=0.18,** $\pi_2(5)$ **=** ***d***
$\xi_3(5)$ **=2.45,** $\omega_3(5)$ **=1.15,** $\pi_3(5)$ **= e**



Main Loop
for $t = 4$

for all links $(i, j) \in A,$ *(1,2), (1,3), (2,3), (2,4), (3,4)*

(1,2)

$$e_1(4) = \sum_{k}^{k_{ij}(t)} \left[\tau_{12}^k(4) * p_{12}^k(4)\right] + \sum_{k}^{k_{ij}(t)} \left[\xi_2\left(4 + \tau_{12}^k(4)\right) * p_{12}^k(4)\right]$$
= (3*0.43)+(4*0.57)+(6.94*0.43)+(6.94*0.57)
= 10.51

$$\gamma_1(4) = \sum_k \left[\rho_{12}^k(4) * \omega_2\left(4 + \tau_{12}^k(4)\right)\right] + \sum_k \left[\rho_{12}^k(4) * \left(\tau_{12}^k(4) + \xi_2\left(4 + \tau_{12}^k(4)\right) - e_1(4)\right)^2\right]$$
= (0.43*0.18) + (0.57*0.18)
+ (0.43*(3+6.94-10.51)^2+0.57*(4+6.94-10.51)^2)
= 0.42

$e_1(4) = 10.51 < \xi_1(4) = \infty$, $\gamma_1(4) = 0.42 < \omega_1(4) = \infty$
Update

$\xi_1(4) = e_1(4)$ =10.51, $\omega_1(4) = \gamma_1(4)$ =0.42, $\pi_1(4) = (1,2)$=*a*

<u>(1, 3)</u>

$$e_1(4) = \sum_k^{k_{ij}(t)} \left[\tau_{13}^k(4) * p_{13}^k(4)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(4 + \tau_{13}^k(4)\right) * p_{13}^k(4)\right]$$

= (1*0.70)+(8*0.30)+(2.45*0.70)+(2.45*0.30)
= 5.55

$$\gamma_1(4) = \sum_k \left[\rho_{13}^k(4) * \omega_3\left(4 + \tau_{13}^k(4)\right)\right] + \sum_k \left[\rho_{13}^k(4) * \left(\tau_{13}^k(4) + \xi_3\left(4 + \tau_{13}^k(4)\right) - e_1(4)\right)^2\right]$$

= (0.70*1.15) + (0.30*1.15)
+ (0.70*$(1+2.45-5.55)^2$+0.30*$(8+2.45-5.55)^2$)
= 11.44

$e_1(4)$ =5.55< $\xi_1(4)$ =10.51, $\gamma_1(4)$ =11.44> $\omega_1(4)$ =0.42

Compare $e_1(4) + 1 * \sqrt{\gamma_1(4)}$ =5.55+3.38 =8.93 <
$\xi_1(4) + 1 * \sqrt{\omega_1(4)}$ =10.51+0.64 =11.15

Update
$\xi_1(4) = e_1(4)$ =5.55, $\omega_1(4) = \gamma_1(4)$ =11.44, $\pi_1(4) = (1,3)$=*b*

<u>(2, 3)</u>

$$e_2(4) = \sum_k^{k_{ij}(t)} \left[\tau_{23}^k(4) * p_{23}^k(4)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(4 + \tau_{23}^k(4)\right) * p_{23}^k(4)\right]$$

= (6*0.90)+(7*0.10)+(2.45*0.90)+(2.45*0.10)
= 8.55

$$\gamma_2(4) = \sum_k \left[\rho_{23}^k(4) * \omega_3\left(4 + \tau_{23}^k(4)\right)\right] + \sum_k \left[\rho_{23}^k(4) * \left(\tau_{23}^k(4) + \xi_3\left(4 + \tau_{23}^k(4)\right) - e_2(4)\right)^2\right]$$

= (0.90*1.15) + (0.10*1.15)
+ (0.90*$(6+2.45-8.55)^2$+0.10*$(7+2.45-8.55)^2$)
= 1.24

$e_2(4)$=8.55< $\xi_2(4) = \infty$, $\gamma_2(4)$ =1.24> $\omega_2(4) = \infty$

Update
$\xi_2(4) = e_2(4)$ =8.55, $\omega_2(4) = \gamma_2(4)$ =1.24, $\pi_2(4) = (2,3)$=c

<u>(2, 4)</u>

$$e_2(4) = \sum_k^{k_{ij}(t)} \left[\tau_{24}^k(4) * p_{24}^k(4)\right] + \sum_k^{k_{ij}(t)} \left[\xi_4\left(4 + \tau_{24}^k(4)\right) * p_{24}^k(4)\right]$$

= (3*0.69)+(1*0.31)+(0*0.69)+(0*0.31)
= 2.38

$$\gamma_2(4) = \sum_k \left[\rho_{24}^k(4) * \omega_4\left(4+\tau_{24}^k(4)\right)\right] + \sum_k \left[\rho_{24}^k(4) * \left(\tau_{24}^k(4) + \xi_4\left(4+\tau_{24}^k(4)\right) - e_2(4)\right)^2\right]$$

= (0.69*0) + (0.31*0)
+ (0.69*(3+0-2.38)^2+0.31*(1+0-2.38)^2)
= 0.856

$e_2(4)$=2.38 < $\xi_2(4)$=8.55, $\gamma_2(4)$=0.856 < $\omega_2(4)$=1.24

Update

$\xi_2(4) = e_2(4)$=2.38, $\omega_2(4) = \gamma_2(4)$=0.865, $\pi_2(4) = (2,4)$=d

<u>(3, 4)</u>

$$e_3(4) = \sum_k^{k_{ij}(t)} \left[\tau_{34}^k(4) * p_{34}^k(4)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(4+\tau_{34}^k(4)\right) * p_{34}^k(4)\right]$$

= (6*0.62)+(8*0.38)+(0*0.62)+(0*0.38)
= 6.76

$$\gamma_3(4) = \sum_k \left[\rho_{34}^k(4) * \omega_4\left(4+\tau_{34}^k(4)\right)\right] + \sum_k \left[\rho_{34}^k(4) * \left(\tau_{34}^k(4) + \xi_4\left(4+\tau_{34}^k(4)\right) - e_3(4)\right)^2\right]$$

= (0.62*0) + (0.38*0)
+ (0.62*(6+0-6.76)^2+0.38*(8+0-6.76)^2)
= 0.94

$e_3(4)$=6.76 < $\xi_3(4) = \infty$, $\gamma_3(4)$=0.94 < $\omega_3(4) = \infty$

Update

$\xi_3(4) = e_3(4)$=6.67, $\omega_3(4) = \gamma_3(4)$=0.94, $\pi_3(4) = (3,4)$=e

$\xi_1(4) = e_1(4)$ **=5.55,** $\omega_1(4) = \gamma_1(4)$ **=11.44,** $\pi_1(4) =$ **(1,3)=*b***
$\xi_2(4) = e_2(4)$ **=2.38,** $\omega_2(4) = \gamma_2(4)$ **=0.865,** $\pi_2(4) =$ **(2,4)=d**
$\xi_3(4) = e_3(4)$ **=6.67,** $\omega_3(4) = \gamma_3(4)$ **=0.94,** $\pi_3(4) =$ **(3,4)=e**

for *t = 3*

for all links $(i, j) \in A$, *(1,2), (1,3), (2,3), (2,4), (3,4)*

<u>(1,2)</u>

$$e_1(3) = \sum_k^{k_{ij}(t)} \left[\tau_{12}^k(3) * p_{12}^k(3)\right] + \sum_k^{k_{ij}(t)} \left[\xi_2\left(3+\tau_{12}^k(3)\right) * p_{12}^k(3)\right]$$

= (2*0.2)+(8*0.8)+(6.94*0.2)+(6.94*0.8)
= 13.74

$$\gamma_1(3) = \sum_k \left[\rho_{12}^k(3) * \omega_2\left(3 + \tau_{12}^k(3)\right)\right] + \sum_k \left[\rho_{12}^k(3) * \left(\tau_{12}^k(3) + \xi_2\left(3 + \tau_{12}^k(3)\right) - e_1(3)\right)^2\right]$$

= (0.2*0.18) + (0.8*0.18)
+ (0.2*(2+6.94-13.74)^2+0.8*(8+6.94-13.74)^2)
= 5.94

$e_1(3)$=13.74< $\xi_1(3) = \infty$, $\gamma_1(3)$=5.94< $\omega_1(3) = \infty$

Update

$\xi_1(3) = e_1(3)$=13.74, $\omega_1(3) = \gamma_1(3)$=5.94, $\pi_1(3) = (1,2)$=*a*

(1, 3)

$$e_1(3) = \sum_k^{k_{ij}(t)} \left[\tau_{13}^k(3) * p_{13}^k(3)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(3 + \tau_{13}^k(3)\right) * p_{13}^k(3)\right]$$

= (7*0.27)+(8*0.73)+(2.45*0.27)+(2.45*0.78)
= 10.30

$$\gamma_1(3) = \sum_k \left[\rho_{13}^k(3) * \omega_3\left(3 + \tau_{13}^k(3)\right)\right] + \sum_k \left[\rho_{13}^k(3) * \left(\tau_{13}^k(3) + \xi_3\left(3 + \tau_{13}^k(3)\right) - e_1(3)\right)^2\right]$$

= (0.27*1.15) + (0.73*1.15)
+ (0.27*(7+2.45-10.30)^2+0.73*(8+2.45-10.30)^2)
= 1.36

$e_1(3)$=10.30< $\xi_1(3)$=13.74, $\gamma_1(3)$=1.36< $\omega_1(3)$=5.94

Update

$\xi_1(3) = e_1(3)$=10.30, $\omega_1(3) = \gamma_1(3)$=1.36, $\pi_1(3) = (1,3)$=*b*

(2, 3)

$$e_2(3) = \sum_k^{k_{ij}(t)} \left[\tau_{23}^k(3) * p_{23}^k(3)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(3 + \tau_{23}^k(3)\right) * p_{23}^k(3)\right]$$

= (5*0.39)+(8*0.61)+(2.45*0.39)+(2.45*0.61)
= 9.28

$$\gamma_2(3) = \sum_k \left[\rho_{23}^k(3) * \omega_3\left(3 + \tau_{23}^k(3)\right)\right] + \sum_k \left[\rho_{23}^k(3) * \left(\tau_{23}^k(3) + \xi_3\left(3 + \tau_{23}^k(3)\right) - e_2(3)\right)^2\right]$$

= (0.39*1.15) + (0.61*1.15)
+ (0.39*(5+2.45-9.28)^2+0.61*(8+2.45-9.28)^2)
= 3.29

$e_2(3)$=3.29< $\xi_2(3) = \infty$, $\gamma_2(3)$=3.29> $\omega_2(3) = \infty$

Update

$\xi_2(3) = e_2(3)$ =9.28, $\omega_2(3) = \gamma_2(3)$ =3.29, $\pi_2(3) = (2, 3)$=c

(2, 4)

$$e_2(3) = \sum_k^{k_{ij}(t)} \left[\tau_{24}^k(3) * p_{24}^k(3)\right] + \sum_k^{k_{ij}(t)} \left[\xi_4\left(3+\tau_{24}^k(3)\right) * p_{24}^k(3)\right]$$

= (9*0.55)+(5*0.45)+(0*0.55)+(0*0.45)

= 7.20

$$\gamma_2(3) = \sum_k \left[\rho_{24}^k(3) * \omega_4\left(3+\tau_{24}^k(3)\right)\right] + \sum_k \left[\rho_{24}^k(3) * \left(\tau_{24}^k(3) + \xi_4\left(3+\tau_{24}^k(3)\right) - e_2(3)\right)^2\right]$$

= (0.55*0) + (0.45*0)

+ $(0.55*(9+0-7.2)^2+0.45*(5+0-7.2)^2)$

= 3.96

$e_2(3)$=7.20 < $\xi_2(3)$=9.28, $\gamma_2(3)$=3.96 > $\omega_2(3)$=3.29

Compare $e_2(3) + 1*\sqrt{\gamma_2(3)}$ =7.20+1.99 =9.19 <

$\xi_2(3) + 1*\sqrt{\omega_2(3)}$ =9.28+1.81 =11.09

Update

$\xi_2(3) = e_2(3)$ =7.20, $\omega_2(3) = \gamma_2(3)$ =3.96, $\pi_2(3) = (2,4)$=d

(3, 4)

$$e_3(3) = \sum_k^{k_{ij}(t)} \left[\tau_{34}^k(3) * p_{34}^k(3)\right] + \sum_k^{k_{ij}(t)} \left[\xi_3\left(3+\tau_{34}^k(3)\right) * p_{34}^k(3)\right]$$

=6.52

$$\gamma_3(3) = \sum_k \left[\rho_{34}^k(3) * \omega_4\left(3+\tau_{34}^k(3)\right)\right] + \sum_k \left[\rho_{34}^k(3) * \left(\tau_{34}^k(3) + \xi_4\left(3+\tau_{34}^k(3)\right) - e_3(3)\right)^2\right]$$

= (0.88*0) + (0.12*0)

+ $(0.88*(7+0-6.52)^2+0.12*(3+0-6.52)^2)$

= 1.69

$e_3(3)$=6.52 < $\xi_3(3) = \infty$, $\gamma_3(3)$=1.69 < $\omega_3(3) = \infty$

Update

$\xi_3(3) = e_3(3)$ =6.52, $\omega_3(3) = \gamma_3(3)$ =1.69, $\pi_3(3) = (3,4)$=e

for t =3

$\xi_1(3)$ **=10.30,** $\omega_1(3)$ **=1.36,** $\pi_1(3) =$ **(1,3)=*b***

$\xi_2(3))$ **=7.20,** $\omega_2(3)$ **=3.96,** $\pi_2(3) =$ **(2,4)=d**

$\xi_3(3)$ **=6.52,** $\omega_3(3)$ **=1.69,** $\pi_3(3) =$ **(3,4)=e**

for *t = 2*

$\xi_1(2)$ **=7.25,** $\omega_1(2)$ **=2.11,** $\pi_1(2) =$ **(1,3)=*b***
$\xi_2(2)$ **=7.05,** $\omega_2(2)$ **=0.048,** $\pi_2(2) =$ **(2,4)=d**
$\xi_3(2)$ **=1.11,** $\omega_3(2)$ **=0.098,** $\pi_3(2) =$ **(3,4)=e**

for *t = 1*

$\xi_1(1)$ **=12.22,** $\omega_1(1)$ **=9.927,** $\pi_1(1) =$ **(1,2)=*a***
$\xi_2(1)$ **=3.04,** $\omega_2(1)$ **=0.998,** $\pi_2(1) =$ **(2,4)=d**
$\xi_3(1)$ **=5.20,** $\omega_3(1)$ **=0.96,** $\pi_3(1) =$ **(3,4)=e**

for *t = 0*

$\xi_1(0)$ **=9.69,** $\omega_1(0)$ **=4.06,** $\pi_1(0) =$ **(1,3)=*b***
$\xi_2(0)$ **=3.79,** $\omega_2(0)$ **=0.17,** $\pi_2(0) =$ **(2,4)=d**
$\xi_3(0)$ **=5.15,** $\omega_3(0)$ **=11.31,** $\pi_3(0) =$ **(3,4)=e**

End

Chapter 6. Computational Testing

In this chapter, the performance of the algorithms, PMM, PMV, PMMV, and TAMMV1 are evaluated through numerical experiments, which are intended to assess the computational performance on randomly generated networks as well as representations of an actual Los Angeles highway network. The objectives of the computational test are multiple:

- Check the validity of the minimum mean-variance model
- Comparison of computational complexity for TAMMV1 and TAMMV2 algorithm
- Apply the model to real world traffic problem

This chapter is organized as follows. Firstly, in Section 6.1, the methodology for generating the networks with their stochastic, time-dependent arc travel times is described and the experimental design is given. In Section 6.2, the results of the tests are presented and analyzed. In Section 6.3, the problem of selecting a "best" route during afternoon rush hour between two points in LA area is used to illustrate the results of the algorithms developed in Chapters 4 and 5. Finally, in Section 6.4, a discussion of the results and conclusions is presented.

6.1 Experimental Design

The experiments described in this chapter are conducted on twenty seven randomly generated networks with randomly generated time-dependent probability distribution functions of the arc travel time random variables. The methodologies to generate these networks and distribution functions are described in Subsections 6.1.1 and 6.1.2, respectively. In Subsection 6.1.3, the design of the experiments is described in detail.

6.1.1 Generating the Networks

A GT Internetwork Topology Models (GT-ITM) is used to randomly generate the networks for these experiments. For each network that is generated, the number of nodes, and the probability of edge between each pair of nodes is specified. All generated networks are directed graphs and no arcs are duplicated.

Since the primary concern of this study is in the application of these algorithms to transportation systems, the networks have been generated such that their average connection probabilities of each node are 0.2(density 2), 0.4(density 4), and 0.6(density 6). The generated network with same connection probability ensures that the networks with the same number of nodes will have nearly the same number of arcs.

6.1.2 Generating the Arc Travel Time Random Variables

Once the topology of a network is specified, the arc weights can be determined. The arc weights are random variables with time-varying probability distribution functions. The probability mass functions of the arc weight random variables are randomly generated, corresponding to either discrete random variables or approximations of continuously distributed random variables. The arc travel times are assumed to be independent across time and space. Their pmfs are generated for each arc at each time interval as follows:

Given the number of elements in the pmfs, P (assumed constant across arcs and departure times):

Step 1: Generate P pairs of scaled uniform random variants. The first random variant of the pair will represent a possible travel time. This random variant is scaled between 1 and 10 units. If the same number is generated, it will be discarded.

Step 2: The second random variant, the probability of the occurrence of such a travel time, will be generated as follows. First, generate P numbers of random numbers between 1 and 100. Find the proportions of each generated value (divide each value by sum of all generated values) so that their sum is equal to 1.

Step 3: Sort the pairs of random variants in ascending order of the first element of the pair (corresponding to increasing travel time).

6.1.3 Design of the Experiments

Four factors must be specified in order to generate the network topology and the pmfs of the arc travel time random variables: the number of nodes, probability of arcs, the duration of travel time intervals, and the number of elements in the; pmf's. (The duration of a time interval is constant over all the tests). The number of elements in the pmfs is nearly constant across arcs and time intervals as explained previously.

Three levels of the number of nodes are considered: 50, 100, and 500 nodes. Also, three levels of arc densities are considered: 2, 4, and 6. Three topological networks of each level are generated, for a total of 27 networks. The time interval size is one time unit in duration and is constant over all the experiments. Three levels of the duration of the peak period are considered: 10, 30, and 60 units of time (time intervals) in size. For example, a peak period of 10 minutes will consist of 10 one minute time intervals and a peak period of 60 minutes will consist of 60 one minute time intervals. Finally, two levels of the constant number of elements in the probability mass functions are considered: 2and 5. This result in 162 different combinations, as every combination of the number of time intervals and number of elements in the pmfs are considered for each of the 27 networks.

The algorithms of TAMMV1 and TAMMV2 determine one path from all origin nodes to a pre-specified destination node for each departure time interval, these algorithms are implemented in C programming language. The experiments are done on Sun UltraSPARC-IIi Workstations (360 MHz of clock speed and 128 MB RAM) running Solaris 8. An Ethernet communication speed of 100Mbits/sec was assumed for the communication of data. This forms a common speed of an Ethernet cable.

The run times of the procedures are recorded. Because three topological networks of each level are tested under same condition (edge probability, time interval and the number of pmfs), we record the average running time for each topological network. The runtime does not include input/output time.

6.2 Experimental Results

The results of the experiments on twenty seven randomly generated networks are summarized in Tables 6.1 through 6.4 in Section 6.2.1. The number of nodes in the networks is indicated by the heading “Nodes”, arc density “ArcD”, the number of time intervals “TI”, and the number of elements in the pmf s by “Prob”. For the networks that can be specified by the same (n, a, t, p) (equivalent “Nodes”, “ArcD”, “TI” and “Prob”) the results are averaged.

6.2.1 Performance on Randomly Generated Networks

In all of the tests, the SA list of each algorithm is implemented as a deque list. A node or node-label pair always enters the SA list at the back although it has been entered in the SA list previously or not. The tests of this chapter are not intended to test the performance of the procedures under a variety of SA list structures.

The actual average running times for the PMV and PMMV procedures are given in Table 6.1 for networks with two sizes of the nodes n=50 and 100, three values of the time interval TI=10, 30, and 60, two values of the PMFs p=2 and 5, and fixed arc density a=4 . The run time of the two algorithms are very similar because algorithmic procedures are almost same except updating rule of node label (for PMV-variance, for PMMV-mean and variance).

Table 6.2 shows the PMMV algorithm running time results for different values of the arc density a=2, 4, and 6 for same number of n, t, and p. To describe the performance of this algorithm, the natural log of the run time, given as RT_{PMMV} for the PMMV algorithms, in CPU milliseconds is regressed against the natural log of the number of nodes(n), arc density(a), number of time intervals(t), and the number of elements in the PMFs(p), resulting in the equations with an R2 value of 94.47%:

$$RT_{PMMV} = (0.0000033) * p^{0.75} * n^{2.09} * a^{1.04} * t^{0.95}$$

This regressing model is provided without testing of large number of nodes. Therefore, it is difficult to apply for networks of large size. This remaining testing job can be done in the future.

Table 6.1. Comparison of run times in c.p.u seconds for PMV and PMMV algorithms

Nodes	TI											
	10				30				60			
	p=2		p=5		p=2		p=5		p=2		p=4	
	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2
50	0.61	0.69	1.27	1.32	1.88	2.06	3.75	3.95	3.77	4.07	7.56	7.97
100	3.00	3.10	5.38	5.53	7.35	7.67	14.62	15.82	13.54	14.53	28.66	29.58

Table 6.2. Run times in c.p.u seconds for PMMV algorithm

Nodes	Arc D.	TI					
		10		30		60	
		Prob=2	Prob=5	Prob=2	Prob=5	Prob=2	Prob=4
50	2	0.55	0.85	1.57	1.89	3.66	4.97
	4	0.78	1.41	2.27	3.66	5.19	8.97
	6	1.03	1.99	2.96	4.85	6.59	12.10
100	2	1.48	2.57	4.18	7.42	9.49	16.87
	4	3.21	5.80	8.14	15.46	17.00	32.11
	6	8.98	13.61	27.34	19.48	29.90	77.04

The average run times for the TAMMV1 and TAMMV2 algorithms are given in Tables 6.3 and 6.4, respectively. The run times refer to the average time for each experiment over all networks with similar (n, a, t, p) representation. For example, the first value 0.162 in Table 6.3, corresponding to n of 50, a 0.2, t of 10, p of 2, gives the average run time (in CPU seconds) resulting from experiments conducted on all three 50 node networks with arc density 2 (which is edge connecting probability of 0.2) with (50, 2, 10, 2). Recall that each experiment is tested with given a starting and a destination node.

Table 6.3. Run times in c.p.u seconds for TAMMV1 algorithm

Nodes	Arc D.	TI					
		10		30		60	
		Prob=2	Prob=5	Prob=2	Prob=5	Prob=2	Prob=4
50	2	0.162	0.45	0.47	1.27	0.95	2.50
	4	0.37	0.94	1.16	2.79	2.16	5.73
	6	0.60	1.50	1.84	4.40	3.45	8.90
100	2	0.75	1.89	2.20	5.56	4.32	11.17
	4	2.62	5.32	6.04	14.10	11.22	27.64
	6	8.04	12.82	14.06	26.55	22.49	48.40
500	2	1433	1460	1512	1803	1700	1990
	4	10100	10390	10540	11532	12105	12785
	6	32236	33009	34592	36476	37041	39069

Table 6.4. Run times in c.p.u seconds for TAMMV2 algorithm

Nodes	Arc D.	TI					
		10		30		60	
		Prob=2	Prob=5	Prob=2	Prob=5	Prob=2	Prob=5
50	2	0.15	0.43	0.45	1.20	0.89	2.30
	4	0.34	0.92	1.08	2.66	2.13	5.30
	6	0.56	1.45	1.69	4.52	3.25	8.18
100	2	0.74	2.81	2.17	5.98	4.78	10.78
	4	3.07	4.67	6.35	12.51	14.89	28.56
	6	5.18	13.18	21.53	27.57	40.11	61.15
500	2	1744	1783	2674	2781	4068	4446
	4	11961	12673	15024	16190	17260	18472
	6	38095	39145	40972	43895	45193	48709

To characterize the performance of two algorithms, the natural log of the run time, given as RT_{TA1} and RT_{TA2} for the TAMMV1 and TAMMV2 algorithms respectively, in CPU milliseconds is regressed against the natural log of the number of nodes(n), arc density(a), number of time intervals(t), and the number of elements in the PMFs(p), resulting in the equations with an R^2 value of 97.50% and 97.89%, respectively:

$$RT_{TA1} = (0.000000001) * p^{0.66} * n^{3.89} * a^{1.86} * t^{0.64}$$

$$RT_{TA2} = (0.000000001) * p^{0.65} * n^{4.05} * a^{1.76} * t^{0.73}$$

From the regression analyses, it is shown that the average run time of the TAMMV1 and the TAMMV2 algorithms increases much worse than linearly with the number of nodes (power of 3.89 and 4.05) in the network, arc density(power of 1.86 and 1.76). The run time increase well than linearly with the number of elements in the PMFs and the number of time intervals in the peak period. In a direct comparison of the average run times of the TAMMV1 and TAMMV2 algorithms shown in Tables 6.5, it appears that the TAMMV1 algorithm is often faster than is the TAMMV2 algorithm for the majority of the larger-size networks, while the TAMMV1 algorithm appears to be faster for the smaller networks and small number of PMFs.

Table 6.5. Running time in CPU seconds for TAMMV1 and TAMMV2 algorithms

		TI											
		10				30				60			
		p= 2		p= 5		p= 2		p= 5		P= 2		p= 5	
Node	Arc D.	TA1	TA2	TA1	TA2	TA1	TA2	TA1	TA2	TA1	TA2	TA1	TA2
50	2	0.162	0.15	0.45	0.43	0.47	0.45	1.27	1.20	0.95	0.89	2.50	2.30
	4	0.37	0.34	0.94	0.92	1.16	1.08	2.79	2.66	2.16	2.13	5.73	5.30
	6	0.60	0.56	1.50	1.45	1.84	1.69	4.40	4.52	3.45	3.25	8.90	8.18
100	2	0.75	0.74	1.89	2.81	2.20	2.17	5.56	5.98	4.32	4.78	11.17	10.78
	4	2.62	3.07	5.32	4.67	6.04	6.35	14.10	12.51	11.22	14.89	27.64	28.56
	6	8.04	5.18	12.82	13.18	14.06	21.53	26.55	27.57	22.49	40.11	48.40	61.15
500	2	1434	1744	1460	1783	1513	2675	1803	2782	1700	4069	1991	4446
	4	10100	11961	10390	12673	10540	15025	11532	16190	12105	17261	12786	18472
	6	32236	38095	33009	39145	34592	40972	36477	43895	37042	45193	39070	48709

Figures 6.1 and 6.2 show the variation of the running time of the algorithms with the network size. The running time increases steeply for larger value of network size in both algorithms. Figures 6.3 and 6.4 illustrate the variation of the running time of the algorithms with the number of arcs. Again, the running time varies increases steeply for larger value of arc size. The running times are increased with exponentially.

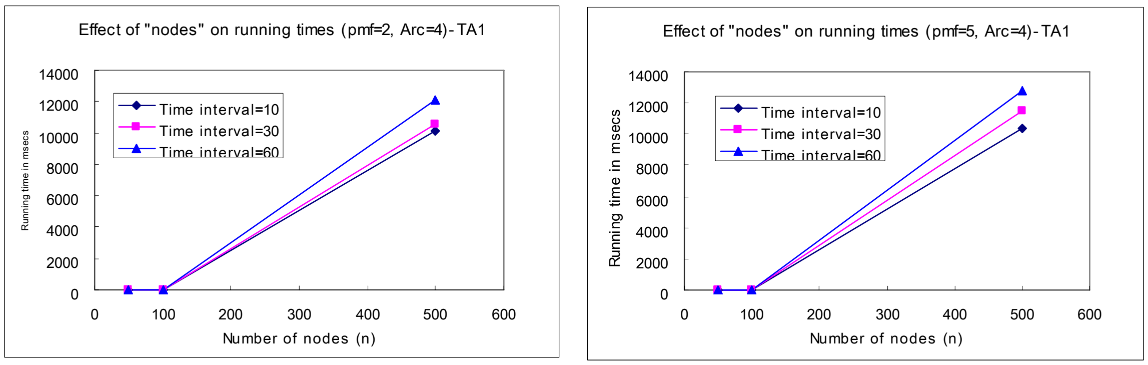


Figure 6.1. Running time of Algorithm TAMMV1 as a function of the number of the nodes, n. Tests are performed for time interval 10, 30, and 60

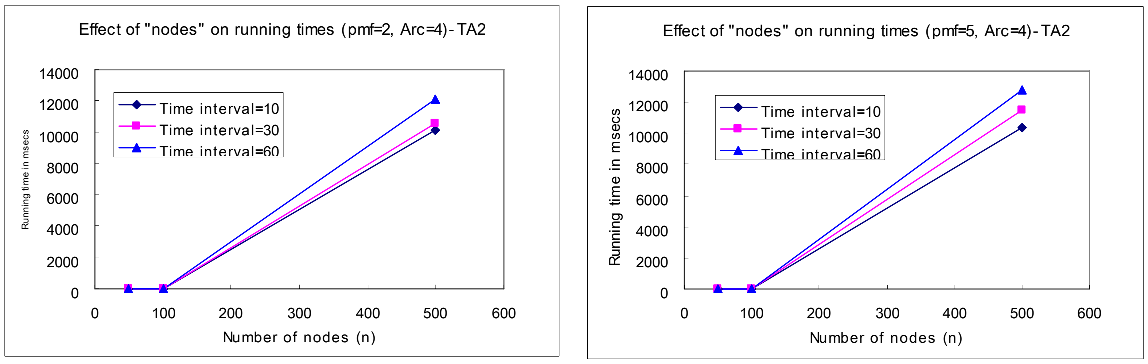
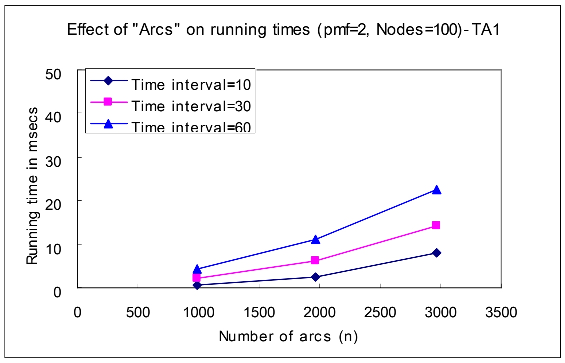


Figure 6.2. Running time of Algorithm TAMMV2 as a function of the number of the nodes, n. Tests are performed for time interval 10, 30, and 60.



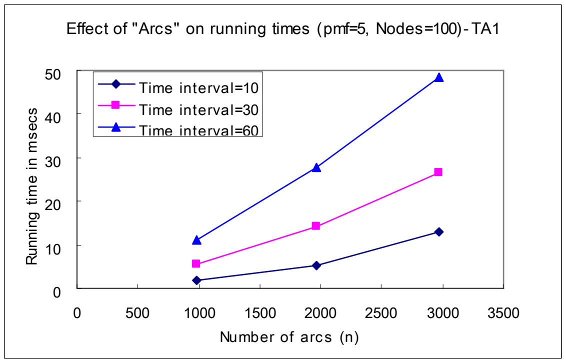
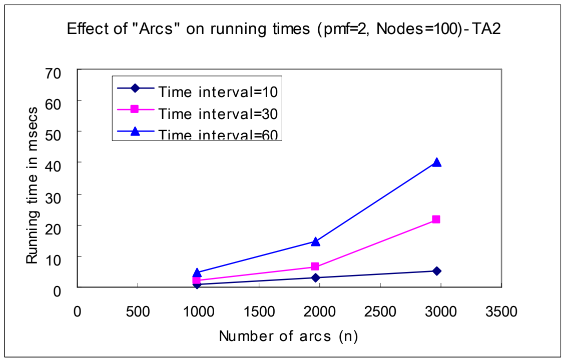


Figure 6.3. Running time of Algorithm TAMMV1 with the number of the arcs, m. Tests are performed for time interval 10, 30, and 60



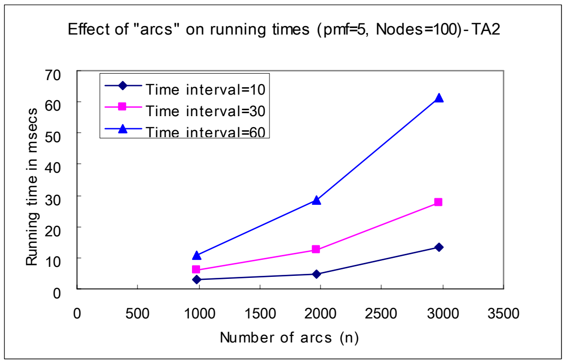
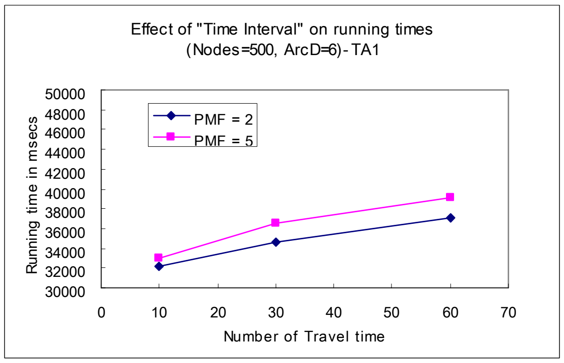


Figure 6.4. Running time of Algorithm TAMMV-STD2 with the number of the arcs, m. Tests are performed for time interval 10, 30, and 60

Figure 6.5 contains the running time results for different values of the time interval t, for n = 500, a = 6. This figure depicts the variation of the running time of Algorithm as a function of the time interval t. The running time increases sharply for smaller values of t, but tends to slow increasing for large values of t.



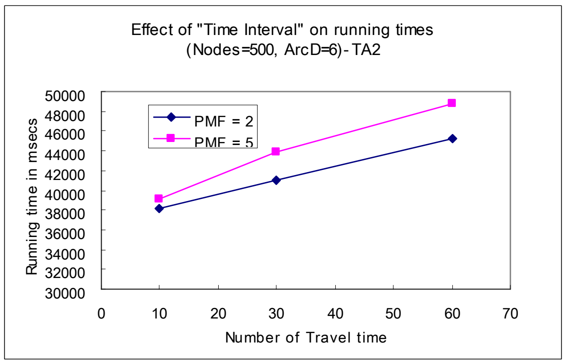


Figure 6.5. Running time of Algorithm TAMMV-STD1and TAMMV-STD 2 as a function of the time Horizon, TI

The tests were run on a Sun UltraSPARC-IIi Workstations which is a server based system with multiple users. In such systems, maintenance activities of the operating system, as well as activities of other users (often many other jobs by other users were run simultaneously with these tests), precludes the possibility of getting very accurate user c.p.u. times. Thus, the exact runs times are not likely to be reproduced, and in many cases, may be overestimated.

6.3 Applications and Extensions

6.3.1 Introduction

Traffic congestion is particularly relevant in urban routing systems. During rush hours in LA, the travel time increases dramatically in most urban areas. This implies that the travel time over a road depends upon the time at which a vehicle travels along the road. Assad (1988) addressed the issue of traffic congestion in travel time determination.

In this chapter, the problem of selecting a "best" route on which to transport hazardous subjects between two specific points in LA area is used to illustrate the results of the algorithms developed in chapter 5. Several problem formulations are addressed, including: (1) determination of a priori least expected time paths (2) determination of a priori minimum variance paths, (3) determination of a priori minimum mean-variance paths, (4) determination of the "best" nest arc routing using previous two algorithms.

The Freeway Performance Measurement System (PeMS) is a joint project between California Department of Transportation and university of California, Berkeley. The intent of this project is to collect historical and real-time freeway data from freeways in the State of California in order to compute freeway performance measures. Because of availability of real time transportation data and high probability of congestion, we select LA urban area to apply our algorithms. In Section 6.3.2, the LA network is described and a specific freeway section is analyzed based on morning and afternoon rush hour. Also, the method to convert speed data to travel time data is presented. In section 6.3.3, the least expected time paths, minimum variance paths, minimum mean-variance paths, and time adaptive minimum mean-variance paths procedures are generated between two given points in LA. The concluding remarks are given in section 6.3.4.

6.3.2 LA area Traffic Data analysis

The traffic sensor is designed for permanent or temporary installation into or onto the road surface for the collection of traffic data. These sensors are geographically distributed and capable of communication and computation. The fixed sensors collect traffic information at the location where they are placed while the sensors on vehicles provide vehicle specific speed and location information etc,. As shown in figure 6.6, the traffic sensors are placed on all major Interstate and highway in LA area. Among the available sensor information, we only need to collect vehicles speed and analyze these speed data to convert travel time data for the given section.

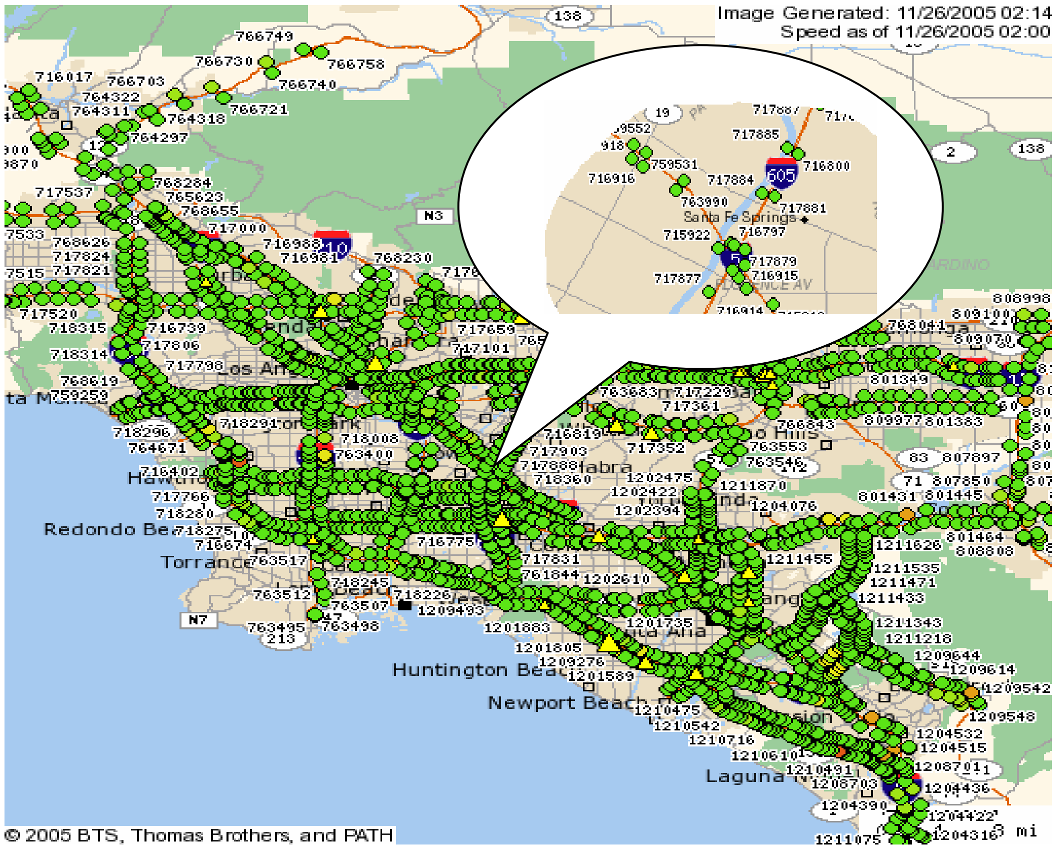


Image Generated: 11/26/2005 02:14
Speed as of 11/26/2005 02:00
Santa Fe Springs
Redondo Beach
Torrance
Huntington Beach
Newport Beach
Laguna
© 2005 BTS, Thomas Brothers, and PATH

Figure 6.6. Major highway sensor location map in LA area

Every five minutes, PeMS aggregate the lane-by-lane 30 second data to compute one number that represents the 5-minute aggregate over all lanes at that vehicle detector station. The results are placed on the FTP site. In order to download the data, a simple client program need to write for periodically grab the data over the Internet from our FTP server. To analyze this sensor data, we select a specific section (from node 24 to 25) and collect data from all sensors on the highway, as shown in figure 6.6. Because the travel time from home to work on a Monday morning could be different from that on a Tuesday morning or Friday morning, we collect the data every five minutes for Monday only.

There are 9 sensors on the highway and read vehicle speed of all 4 lanes. Figures 6.8a through 6.8d shows this result, and its average speed is given in figures 6.9a and 6.9b. As you can see the figures 6.8 and 6.9, there are two congestions, morning and afternoon, on this arc 24-25. These figures show that the duration of morning rush hour is around one hour start form about 5:50am end 7:00am and the afternoon rush hour start around 4:00pm end around 8:00pm. Also, during afternoon rush hours in this section, the traffic congestions are much severe and travel time of this highway section increases dramatically.

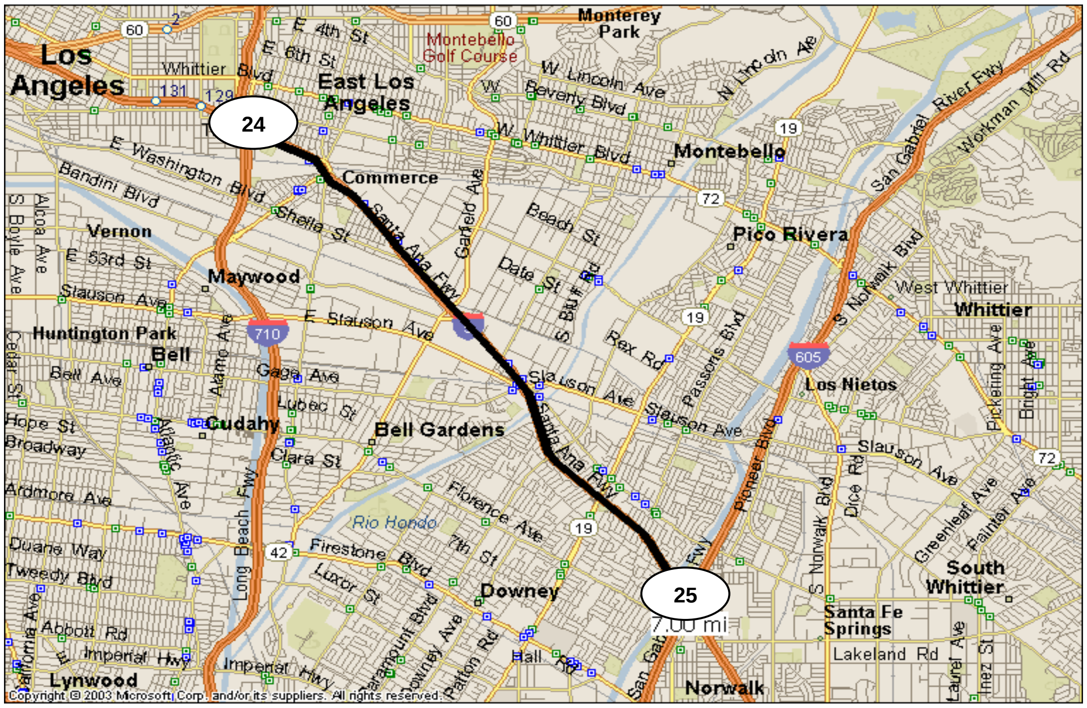


Figure 6.7. A section 24-25 in Interstate highway 5S

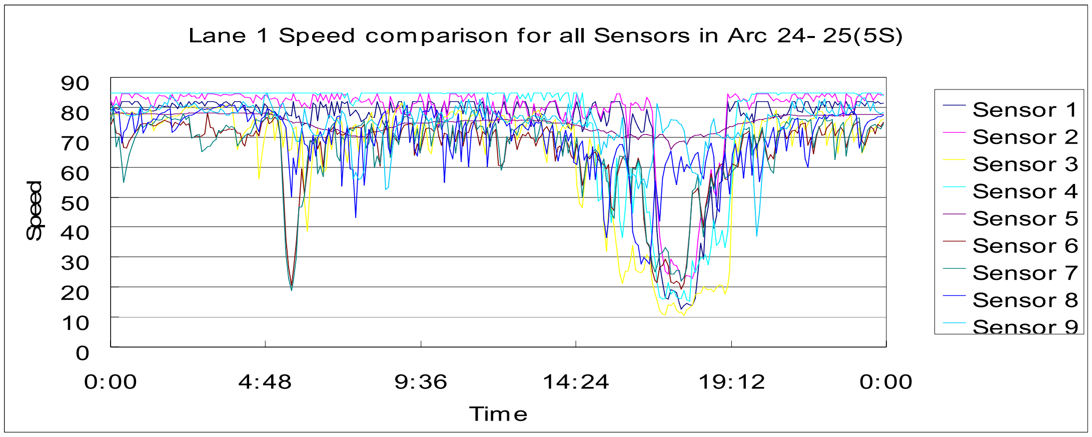


Figure 6.8a. Lane 1 speed for all sensors in arc 24-25

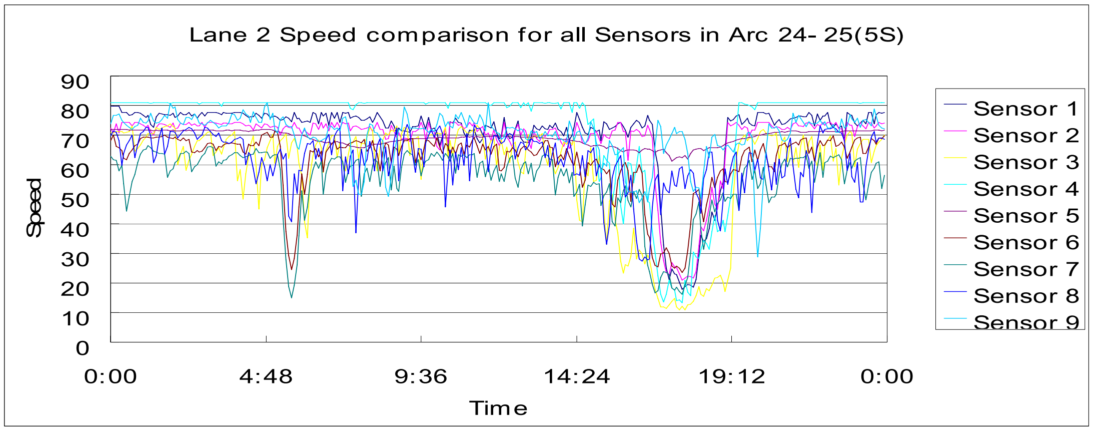


Figure 6.8b. Lane 2 speed for all sensors in arc 24-25

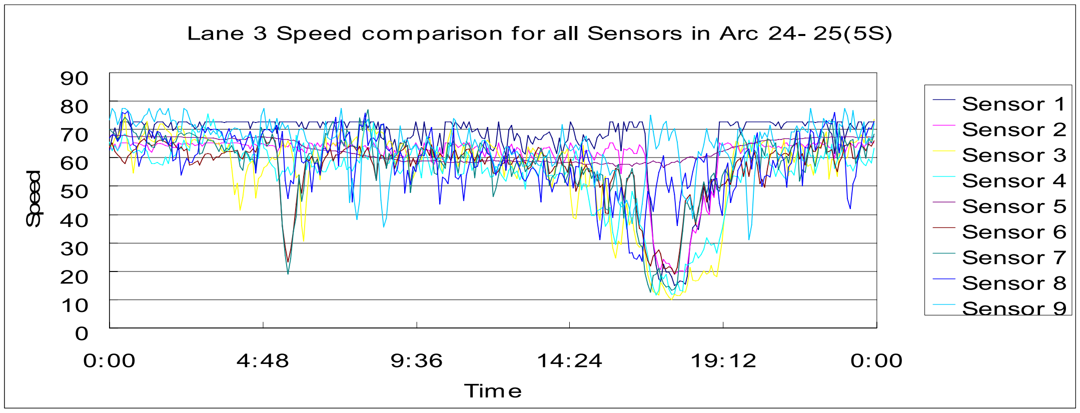


Figure 6.8c. Lane 3 speed for all sensors in arc 24-25

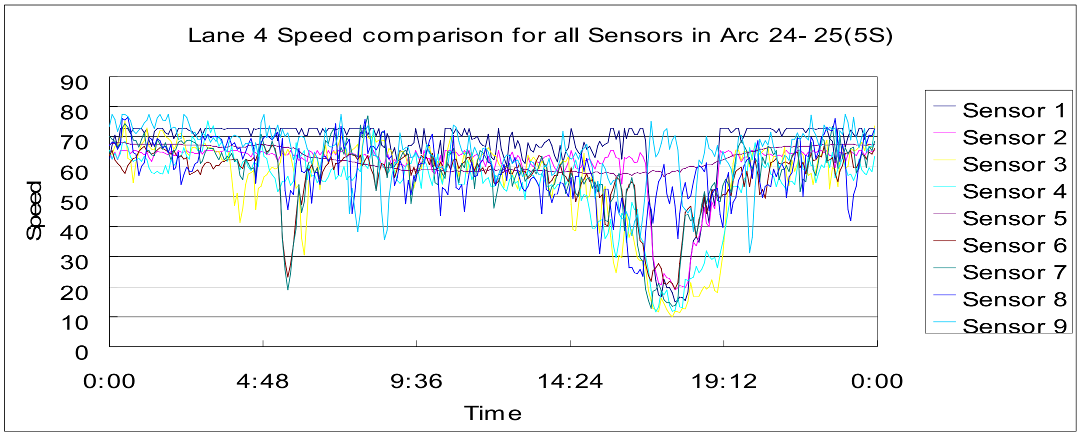


Figure 6.8d. Lane 4 speed for all sensors in arc 24-25

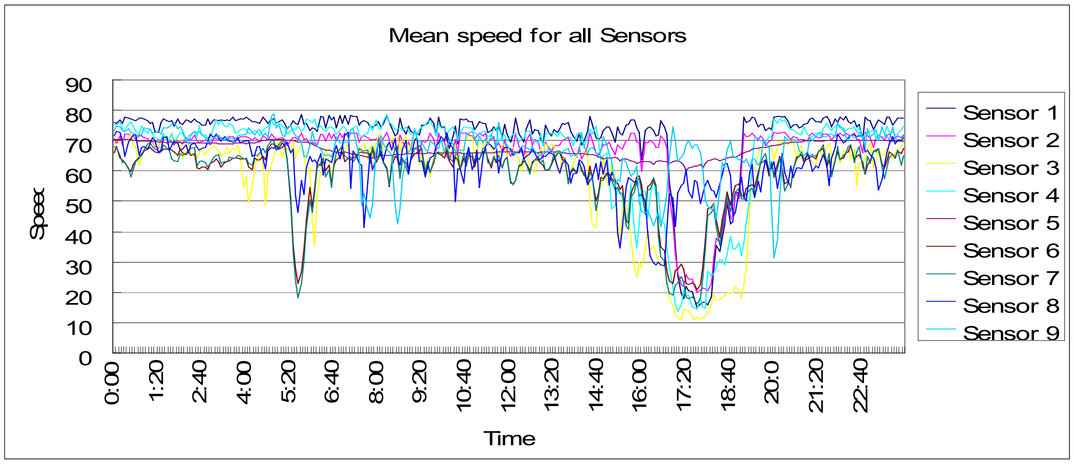


Figure 6.9a. Mean speed for all sensors in arc 24-25

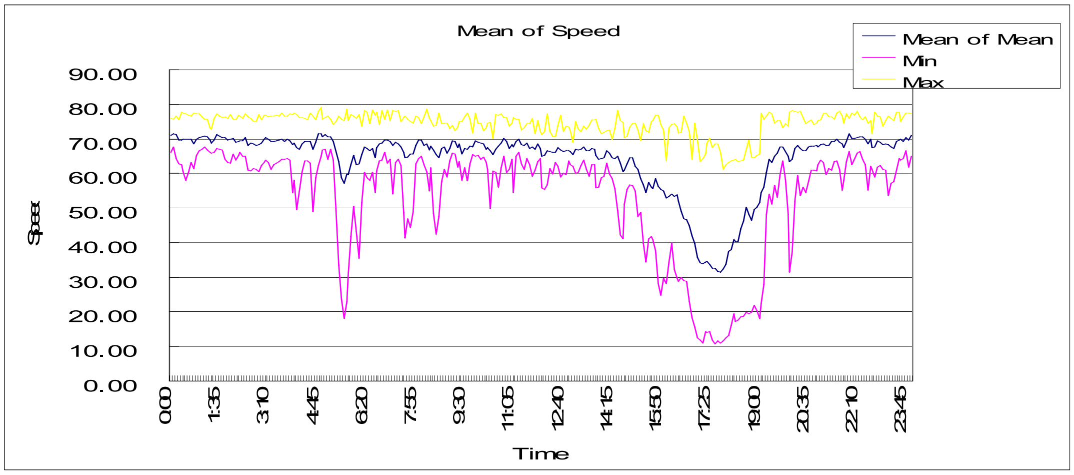


Figure 6.9b. Mean of all 4 lanes speed for all sensors in arc 24-25

This speed data can be transformed to travel time data with distance information.

Since the travel times can be treated as random variables whose probability distribution functions vary with time, probability distribution functions were constructed using these travel time data. Therefore, we converted this every 5 minutes data to every 30 minutes data, as shown in figure 6.11. If the travel time accordance is less than 5% then it can be combined with nearest neighbor travel time. For example, if the frequency of travel time 7 for arc *a1* is 141 out of 145 and the frequency of travel time 8 for arc *a1* is 4 out of 145, the probability of being travel time 8 is only 4/145 = 2.7%, so we can assume the travel time of arc a1 is 7 with 100%. Table 6.6 shows the example of PMFs for this arc.

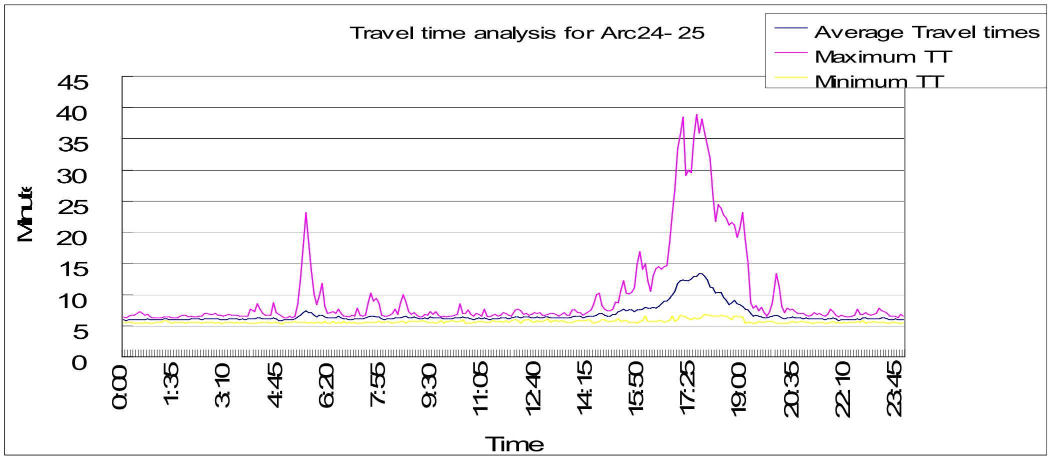


Figure 6.10. Travel time from node 24 to 25

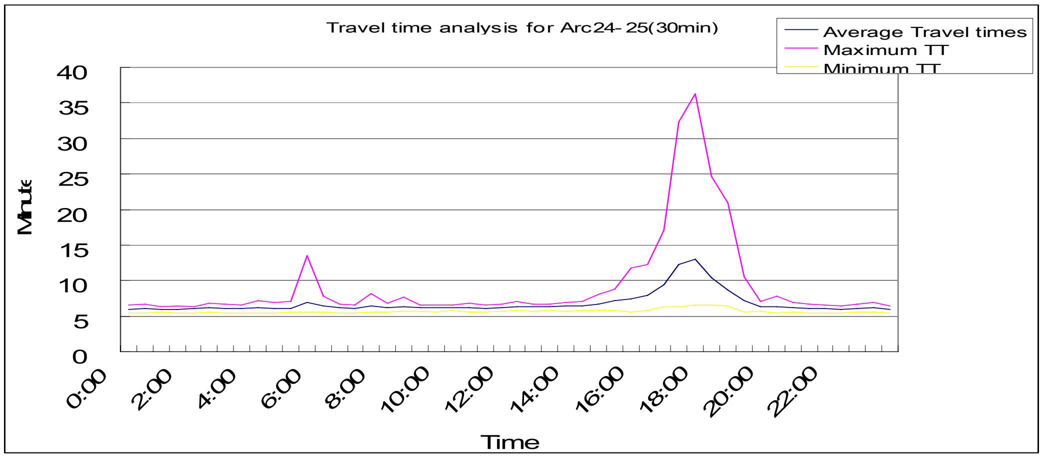


Figure 6.11. Travel time data for every 30 minutes for arc 24-25.

Table 6.6. Example of PMFs for the arc *a1*.

Time interval	Travel time	Probability
16:00 ~ 16:30	7	0.06
	8	0.15
	9	0.29
	10	0.21
	16	0.29
16:30 ~ 17:00	7	0.21
	9	0.17
	12	0.39
	18	0.23
17:00 ~ 17:30	10	0.11
	14	0.19
	20	0.33
	26	0.37

6.3.3 Problem Description

The Los Angeles highway traffic network is pictured in Figure 6.12. This highway system is represented by a graph with 31 nodes (representing intersections in LA) 53 arcs (representing links of highways between intersections). As shown in previous section, this area has serious traffic congestion problems during afternoon rush hours. Therefore the primary focus of this study is apply the four routing algorithms, PMM, PMV, PMMV, and TAMMV1, to assist commuters (passengers, emergency and commercial vehicles) in making the best decisions on route selection. The routing analysis is based on minimizing two attributes: (a) expected travel time, and (b) Variance of the travel time.

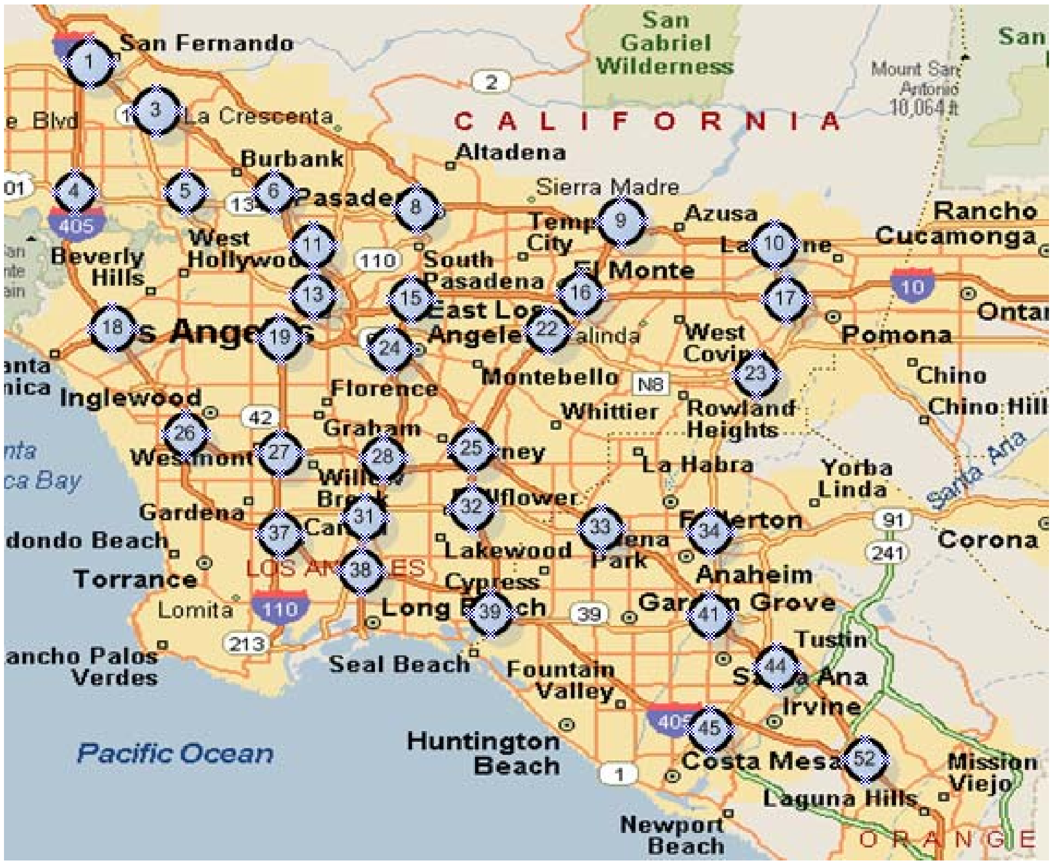


Figure 6.12. The LA traffic Network

In this case study, the starting node and destination node are given: starting node is 4, destination node is 41. No waiting times are allowed during the trip. The trip is scheduled to start from node 4 at 4:00p.m, which is close to the start of rush hour. In Figure 6.11, it was shown that the afternoon peak period lasts for approximately 4 hours on the arcs emanating from each of the nodes. Each routing policy finds the best next node and path information based on the mean or variance or mean-variance of travel time.

The travel times along a route are random variables and vehicle speeds are collected by the traffic sensors on the road. The eleven Mondays (from February 1 to April 31, 2005, except Feb. 21) data were collected for this study. The data contains all lanes speed for every 5 minutes. In many cases, the first lane reserved as a "HOV" lane during the rush hour. Therefore this lane data were discarded to obtain the average lane speed on the specific sensor. Converted travel time data for every arc were generated after averaging vehicle speed with distance of each arc. Probability distribution functions of each arc were generated by EXCEL Macro and stored as a input data file. Detailed descriptions are in previous section.

6.3.4 Results Of The Case Study

A priori minimum expected path: PMM aalgorithm

Using the PMM algorithm, three non-dominated paths out of ten paths are determined:

Path 1: 4-18-26-37-38-39-41

Path 3: 4-18-19-24-25-32-33-41

Path 4: 4-18-19-27-37-38-39-41

The minimum mean travel time and paths information for the given departure times are shown in Table 6.7 and Figure 6.13.

Table 6.7. Minimum mean travel time and paths for MMV algorithm

Departure time	Mean	Next Node	Path
0	73.25795	18	1
30	79.23531	18	1
60	81.14811	18	1
90	83.11905	18	4
120	78.12215	18	3
150	73.38336	18	3
180	71.84217	18	3

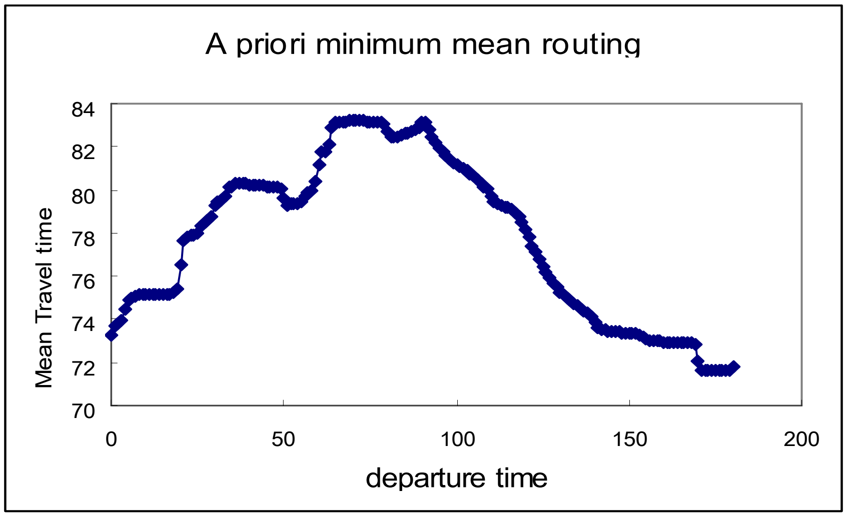


Figure 6.13. Minimum mean travel times for departure time varying

For 180 time interval, 35% for path 1, 50% for path 3, and 15% for path 4 were selected for the minimum mean travel time path from node 4 to node 41. The path 1 was selected for most of the early time interval. If traveler start trip after 5pm to 5:30pm, the path4 became the preferred path. After 5:30pm, path3 is the minimum mean path.

A priori minimum Variance path: PMV algorithm

Using the PMv algorithm, four non-dominated paths are determined:

Path 6: 4-18-19-24-28-31-32-33-41

Path 1: 4-18-26-37-38-39-41

Path 3: 4-18-19-24-25-32-33-41

Path 4: 4-18-19-27-37-38-39-41

The minimum variance and paths information for the given departure times are shown in Table 6.8 and Figure 6.14.

Table 6.8. Minimum variance and paths for PMV algorithm

Departure time	Variance	Next Node	Path
0	46.669945	18	6
30	92.095512	18	1
60	81.028198	18	4
90	66.332283	18	3
120	47.764866	18	3
150	53.012543	18	3
180	53.99812	18	3

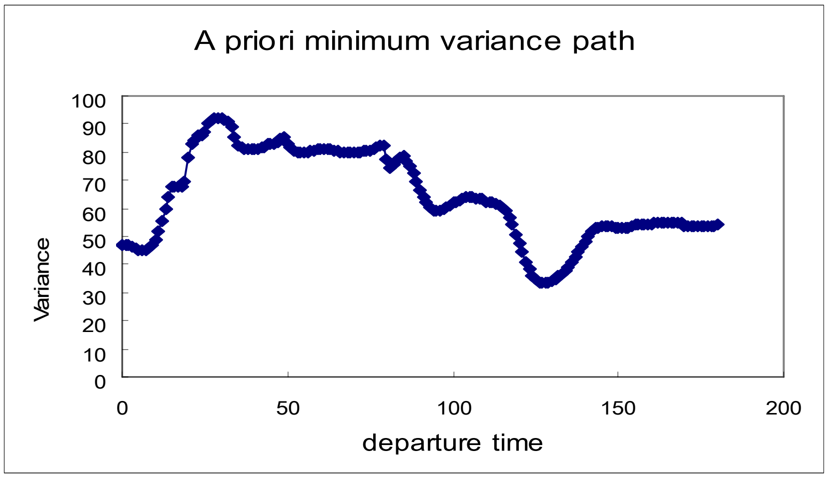


Figure 6.14. Minimum variance of travel times for departure time varying

For 180 time interval, 20% for path 1, 8% for path 6, 53% for path 3, and 19% for path 4 were selected for the minimum variance travel time path from node 4 to node 41. The path 6 was selected for most of the early time interval. The path 3 became the preferred path after 5:30pm.

A priori minimum Mean-Variance path: PMV algorithm

Using the PMv algorithm, two non-dominated paths are determined:

Path 1: 4-18-26-37-38-39-41

Path 3: 4-18-19-24-25-32-33-41

The minimum mean-variance paths information for the given departure times are shown in Table 6.9 and Figure 6.15.

Table 6.9. Minimum mean, variance and paths for PMMV algorithm

Departure time	Mean	Variance	Next Node	Path
0	73.25795	74.955673	18	1
30	79.235306	92.095512	18	1
60	81.148109	107.00051	18	1
90	83.119049	66.332283	18	3
120	78.122154	47.764866	18	3
150	73.383362	53.012543	18	3
180	71.842171	53.998116	18	3

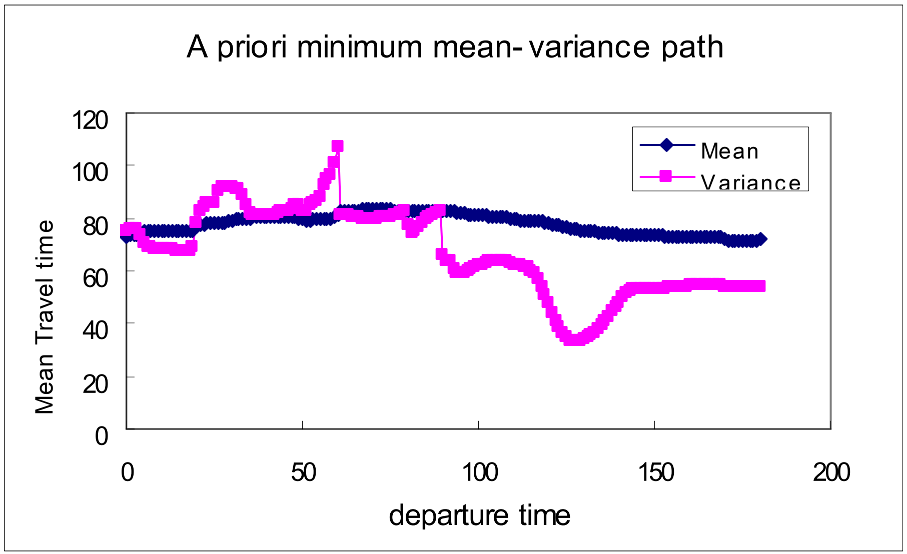


Figure 6.15. Minimum mean-variance for departure time varying

For 180 time interval, 34% for path 1, 50% for path 3, and 16% for path 4 were selected for the minimum mean travel time path from node 4 to node 41. This result was similar with minimum mean path output because expected travel time is the primary decision criterion for selecting the best path. Again, the path 3 was selected for 50% of time interval.

Time-adaptive Mean-Variance path: TAMMV1 algorithm

The resulting hyperpaths for this problem were given in the form of a tree as shown in Figure 6.13. The result was shown that next best node from starting node 4 was node 18 for all travel time interval. The time adaptive minimum mean-variance routing information for the given departure times are shown in Table 6.10 and Figure 6.16.

Table

Table 6.10. Minimum mean, variance and next node for TAMMV1

Departure time	Mean	Variance	Next Node
0	65.000572	51.88723	18
30	67.173195	66.133377	18
60	67.817574	72.658424	18
90	69.390167	76.070457	18
120	67.76239	71.286751	18
150	63.138004	63.080559	18
180	61.665833	67.629845	18

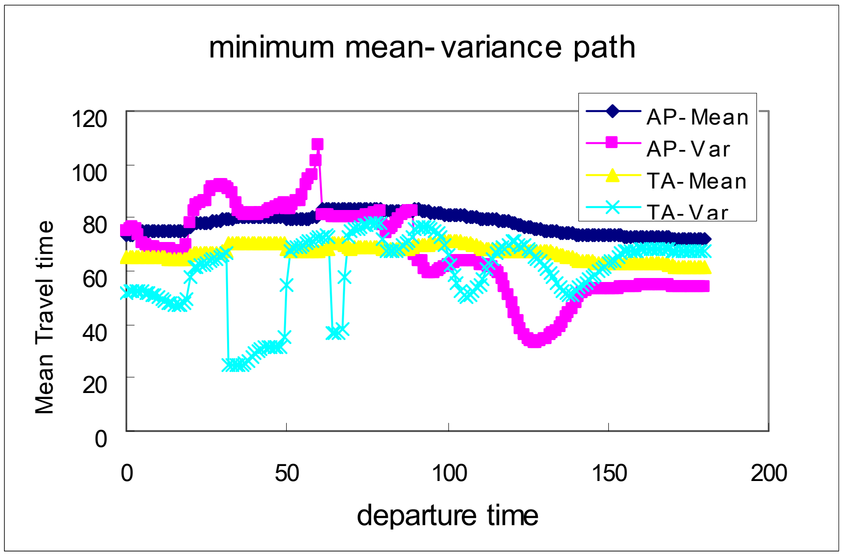


Figure 6.16. Minimum mean-variance for departure time varying

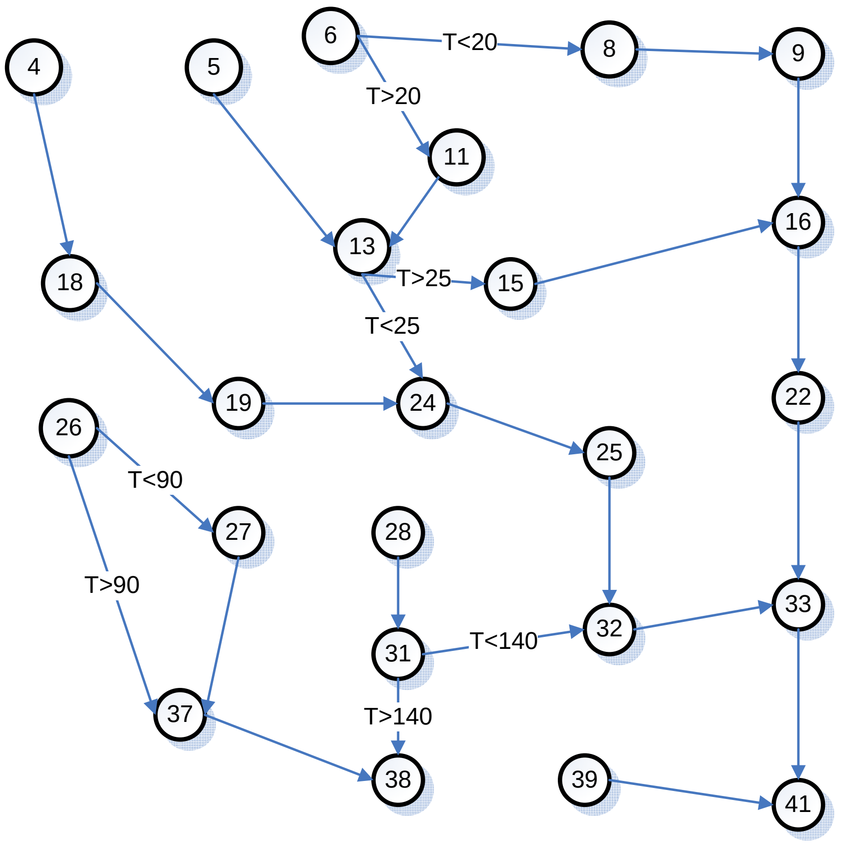


Figure 6.17. Resulting hyperpaths as shown through conditional tree structure

Finally, the path 3 (4-18-19-24-25-32-33-41) is selected as a best compromise path during afternoon rush hours from 4:00 to 7:00pm, considering both travel time and variance.

6.4 Conclusions

In this chapter, the computational tests on randomly generated networks were conducted to assess and compare the average performance of four algorithms, PMV, PMMV, TAMMV1, and TAMMV2. In a direct comparison of the average run times of the TAMMV1and TAMMV2 algorithms, it appeared that the TAMMV2 algorithm was often faster than TAMMV1 algorithm for the small networks, while the TAMMV1 algorithm appeared to be faster for the larger networks. For very large networks, or dense networks, the number of paths that may be examined can grow quite large, and, thus, PMV and PMMV algorithm may perform rather poorly. These two algorithms' worst-case computational complexity is non-polynomial. Therefore, these time adaptive routing algorithms, TAMMV1, and TAMMV2, are more applicable to stochastic time-dependent network problems.

The problem of selecting a "best" routing on which to travel between two points in Los Angeles area in California is used to illustrate the results of the algorithms. The LA network is described and a specific freeway section is analyzed based on morning and afternoon rush hour. Four algorithms are applied to find the minimum expected time paths, minimum variance paths, minimum mean-variance paths, and time adaptive minimum mean-variance paths.

The LA area traffic routing problem is only one of numerous applications for which the procedures of all developed algorithms in this study. Other applications include routing of emergency vehicles to (or from) the scene of a medical emergency, fire fighters to a fire, police officers to a request for service or scene of a crime, commercial trucks to pickups and deliveries, service vehicles to downed power lines, or wreckage from a natural disaster, as well as military applications, and other applications where response time is critical.

Chapter 7. Conclusions and Future Research

7.1 Research Summary

Transportation is a critical component of our lives. Electrical networks bring lights, national highway networks cross distances, manufacturing and distribution networks allow access to consumer products, and computer networks share information globally. In all networking situations we move some entity from one point to another through path as efficiently as possible. Routing (finding path) problems have broad applications in transportation engineering, computer science, operations research, and neurophysiology. They are of importance for passenger and goods movement, message delivery, and more general system control.

Travel time between an origin and destination is often the primary criterion in optimally routing vehicles such as ambulances, police cars, vehicles carrying hazardous substances. Travel times in congested transportation networks are naturally time-dependent and stochastic in nature. In order to optimally route vehicles, both the stochastic and time-dependent nature of the travel times must be considered. Future travel times can be treated as random variables whose probability distribution functions vary with time.

Previous approaches in stochastic time-dependent problems do not account for the fact that travelers often incorporate travel-time variability in decision making. Thus, a route with lower travel-time variability is preferred at certain situations like hazardous material shipment, even if such a route is not one with the lowest mean of travel-time. We recognize the fact that travelers choice not necessarily depend on the least expected time path (LET) but also consider the time variability while choosing a path during the planning stage. This approach is referred in this work as the mean-variance model in which the choice of a route is based on the mean as well as the variance of the path's travel-time. In the current work a methodology for minimum variance and minimum mean-variance path within a route guidance model is presented.

In general stochastic time-dependent networks, two types of routing policies are used for routing in networks: a priori "best" path routing policy and time-adaptive routing policy. For the priori best path routing problem, two algorithms, PMV(a priori minimum variance algorithm) and PMMV(a priori minimum mean-variance algorithm) were developed for determining a minimum variance path and minimum mean-variance path. In both these routing methods it was assumed that drivers use the same path that corresponds the minimum variance or minimum mean-variance to their destination node depending on their actual departure time at an origin node. We found the recursive relationship between means and variances of a given routing policy starting from two adjacent nodes. The node labels are updated by using the recursive formulation. At termination of either algorithm, the final node labels are the minimum variance path and minimum mean-variance path from each node to the destination node for all departure time.

The PMV and PMMV procedures are both specialized modified label correcting algorithms for determining "preferred" paths in stochastic time-dependent networks from all origins to a selected destination, for all departure times in the peak period. Both algorithms are similar because both mean and variance calculations are required in both procedures. The multiple vector labels are required, each containing the variance or mean-variance of path travel time for each departure time. These labels are growing exponentially with network size, resulting in nonpolynomial worst-case performance.

Extensions of these algorithms for determining paths in a time-adaptive routing where a driver is permitted to react to revealed information such as arrival time at intermediate nodes were also discussed. Rather than selecting a priori single best path before travel begins, routes with minimum mean-variance were obtained by allowing the driver to react en route to revealed (actual) arrival times. Two computationally efficient algorithms, TAMMV1(time-adaptive minimum mean-variance algorithm1) and TAMMV2(time adaptive minimum mean-variance algorithm2) presented for determining minimum mean-variance travel time path for all origins to a single destination in a networks where the arc weights were discrete random variables whose probability distribution functions varied a priori minimum variance algorithm with time. At termination of the algorithm, efficient solutions (or non-dominated solutions) were generated. The research proposes that such efficient solutions can be presented to the traveler, who may then make the appropriate choice.

The performance of the algorithms, PMV, PMMV, TAMMV1 and TAMMV2 were evaluated through numerical experiments, which were intended to assess the computational performance on randomly generated networks. The results of these tests showed that the TAMMV1 algorithm is often faster than is the TAMMV2 algorithm for the majority of the larger-size networks, while the TAMMV2 algorithm appears to be faster for the smaller networks. It was shown that while the TAMMV1 algorithm outperforms the TAMMV2 algorithm in dense networks (such as data networks) the TAMMV2 algorithm often outperforms the TAMMV1 algorithm in sparse networks (such as transportation networks).

The proposed algorithms were shown to perform successfully in real-life network of best path between Beverly Hills and Garden Grove in LA. The data used for this purpose was real-time data obtained from California DOT.

7.2 Future Research Directions

In this research, several assumptions are made to solve the stochastic time-dependent network problems. First, the travel times are discredited into small time increment. For more realistic approach, continuous-time framework should be considered. The travel times of arcs are assumed as discrete random variables. Such discrete representations of continuous random variables can result in wrong path selections. Therefore, the solution algorithms need to develop for continuous-time dynamic network optimization problem. Second, waiting times are not allowing during the trip. However, most travelers are very flexible for their trip. To more accurately represent traffic or data network, waiting time should not have the limitation.

The computational tests on this study provide valuable insights into the problem for the first time. However, since the research is still in a very early stage, many interesting tests are not performed and are desired for future research. Specifically, more tests on a real-world network with actual data is recommended. Such tests can provide researchers with an idea how the model and algorithms will perform in a realistic network.

Reliability is another important criterion besides expected travel time and variance, when traveler makes routing decision in stochastic networks. With the real-time traffic sensor information, routing decisions can be made in order to help commuters to minimize delay. This minimum delay (on-time arrival) routing problem can be solved by minimization of linear combination of expected travel time, expected early arrival , and expected late arrival.

In communication and other networks, node failures may be common. In this study, node failures are not explicitly considered. The determination of "best" paths given the probability of node failures is an area for future research. Some of the insight gained though the development of this work may be useful for developing procedures for computing the pdf of the minimum time, or least expected time, between two nodes in a stochastic, time-dependent network.

The constrained optimal routing problems in stochastic time-dependent networks are another area of the future research. With the time window constraints, travelers need to arrive some specific point within given time window during the routing. These routing problems have enormous real-world applications.

References

Adlakha, V.G., (1986), "An Improved Conditional Monte Carlo Technique for Stochastic Shortest Route Problem", *Management Science,* 32, 10, 1860-1367.

Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., (1993), *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Alexopoulos, C., (1997), "State Space Partitioning Methods for Stochastic Shortest Path Problem," *Networks*, 30, 9-21.

Andreatta, G., Romeo, L., (1988), "Stochastic Shortest Paths with Recourse," *Networks*, 18, 193-204.

Bard, J. F., and Bennett, J. E., (1991), "Arc Reduction and Path Preference in Stochastic Acyclic Networks", *Management Science,* 37, 198-215.

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

Bellman, R., (1958), "On a routing problem", *Quarterly of Applied Mathematics*, 16, 87-90.

Bereanu, B., (1966), "On Stochastic Linear Programming: The Laplace transform of the distribution of the optimum and applications", *Journal of Mathematical Analysis and Application,* 15, 280-294.

Bertsekas, D. P., (2000), *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, Massachusetts, 2nd edition.

Bertsekas, D.P., and Tsitsiklis, J.N., (1991), "An analysis of Stochastic Shortest Path Problems", *Mathematics of Operations Research,* 16, 3, 580-595.

Blue, V.J., Adler, J.L., and List, G.F., (1997), "Real-time multiple-objective path search for in vehicle route guidance systems", *Transportation Research Records*, 1588, 10-17.

Brumbaugh-Smith, J. and Shier, D., (1989), "An Empirical Investigation of Some Bicriterion Shortest Path Algorithms", *European Journal of Operational Research*, 43, 2 16-224.

Cai, X., Kloks, T., and Wong, C. K., (1997), "Time-Varying Shortest Path Problems with Constraints", *Networks*, 29, 141-149.

Chabini, I. (1998), "Discrete Dynamic Shortest Path Problems in Transportation Applications: Complexity and Algorithms with Optimal Run Time". *Transportation Research Record 1645,* 170-175.

Chabini, I., (1997), "A new algorithm for shortest paths in discrete dynamic networks", *Proceedings of the 8th IFAC Symposium on Transport systems*, China, Greece, 551-556

Chabini, I., (2001), "Algorithms for k-shortest paths and other routing problems in time-dependent networks", NSF report.

Cheung, R. K., (1998), "Iterative methods for dynamic stochastic shortest path problems," *Naval Research Logistics*, 45, 769–789.

Climaco, J.C. N. and Martins, E.Q.V., (1982), "A Bicriterion Shortest Path Algorithm," *European Journal of Operational Research*, 11, 399-404.

Cook, K. L., and E. Halsey, (1969), "The Shortest Route Through a Network with Time-Dependent Internodal Transit Times', *Journal of Mathematical Analysis and Applications* 14, 492-498.

Corea, G. A., and Kulkarni, V. G., (1993), "Shortest paths in stochastic networks with discrete arc lengths", *Networks,* 23, 175-183.

Corley, H.W. and Moon, I.D., (1985), “Shortest Paths in Networks with Vector Weights,” *Journal of Optimization Theory and Applications*, 46, 79-86.

Croucher, J., (1978), “A note on the stochastic shortest-route problem,” *Naval Research Logistics Quarterly*, 25, 729–732.

Current, J., Marsh, M., (1993), “Multiobjective transportation network design and routing problems: Taxonomy and annotation”, *European Journal of Operational Research*, 65, 4-19.

Dial, B.R., (1969), “Algorithm 360: Shortest Path Forest with Topological Ordering”, *Journal of the Association for Computing Machinery*, 12, 632-633.

Dial, R.B., (1996), “Bicriterion Traffic Assignment: Basic Theory and Elementary Algorithms”, *Transportation Science*, 30, 2, 93-111.

Dijkstra, E. W., (1959), “A note on two problems in connection with graphs”, *Numeriche Mathematics,* 1, 269-271.

Dreyfus, (1969), "An Appraisal of Some Shortest-Path Algorithms", *Operations Research*, 17, 395-412.

Erkut, E. and V. Verter (1995), "Hazardous Materials Logistics," in Facility Location: A Survey of Applications and Methods, a Springer-Verlag book edited by Zvi Drezner.

Eubank, J. B., and Kumin, H. J., (1974), “A method for the solution of the distribution problem of stochastic linear programming”, *SIAM Journal Applied Mathematics,* 26, 225-238.

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

Fox, B.L., (1973), “Calculating kth shortest paths,” INFOR-Canada, *Journal of Operational Research and Information Processing*, 11, 66-70.

Fox, B.L., (1975), “More on kth shortest paths,” *Communications of the ACM,* 18, 279.

Fox, B.L., (1978), “Data Structures and Computer Science Techniques in Operations Research”, *Operations Research*, 26, 686-717.

Frank, H., (1969), “Shortest paths in probabilistic graphs”, *Operations Research,* 17, 83-599.

Fu, L., and Rilett, L.R., (1998), “Expected Shortest Paths in Dynamic and Stochastic Traffic Networks”, *Transportation Research-Part B,* 32, 7, 499-511.

Gao, S., and Chabini, I., (2001), “The best routing policy problem in a stochastic time-dependent network”, NSF report.

Glover, F., Glover, R., and Klingman, D., (1984), “Computational study of an improved shortest paths algorithm”, *Network,* 14, 25-37.

Goldberg, A. V., Radzik, T., (1993), “A heuristic improvement of the Bellman-Ford algorithm”, *Applied Mathematical Letter,* 6, 3-6.

Hagstrom, J., (1990), “Computing the Probability Distribution of Project Duration in a PERT Network”, *Networks*, 10, 231-244.

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

Hansen, P., (1980), “Bicriterion Path Problems, In G. Fandel and T. Gal (Eds): Multiple Criteria Decision Making”, *Springer*, Berlin, 109-127.

Hansler, E., (1972), “A fast recursive algorithm to calculate the reliability of communication network”, *IEEE Transportation Communications,* 20, 637-640.

Hayhurst, G. B., and Shier, D. R., (1991), “A factoring approach for the Stochastic Shortest Path Problem”, *Operations Research Letters,* 10, 329-334.

Henig, M. I., (1990), "Risk Criteria in a Stochastic Knapsack Problem", *Operations Research,* 38, 820-825.

Henig, M.I., (1985), “The Shortest Path Problem with Two Objective Functions,” *European Journal of Operations Research*, 25, 281-291.

Henig, M.I., (1994), “Efficient Interactive Methods for a Class of Multiattribute Shortest Path Problems,” *Management Science*, 40, 7, 891-897.

Hoffman, W. and Pavley, R., (1959), “A method for the solution of the n-th best path problem,” *Journal of the Association for Computing Machinery*, 6, 506-514.

Ishii, H., Shiod, S., Nishtida, T., and Namasuya, Y., (1981), “Stochastic spanning tree problem”, *Discrete Applied Mathematics,* 3, 263-273.

Kalbfleisch, J., (1985), *Probability and Statistical Inference: Volume 1: Probability*, Springer-Verlag, New York, Chapter 5.

Kamburowski, J., (1985), "A Note on the Stochastic Shortest Route Problem", *Operations Research*, 33, 696-698.

Kaufman, D. E., and Smith, R. J., (1993), “Fastest paths in time-dependent networks for IVHS application”, *IVHS Journal,* 1, 1-11.

Kulkarni, V. G., (1986), “Shortest Paths in networks with exponentially distributed arc lengths”, *Networks,* 16, 255-274.

Lawler, E. L., (1976), “Combinatorial Optimization: Networks and Matroids, Holt, Rinehart & Winston, New York, NY.

Lawler, E.L., (1972), “A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem,” *Management Science*, 18, 401-405.

Lawler, E.L., (1977), "Comment on computing the k shortest path in a graph," *Communications of the ACM*, 20, 603-604.

List, G. F., Mirchandani, P.B., Turnquist, M. A., Zografos K. G., (1991), "Modeling and analysis for hazardous materials transportation: Risk analysis, routing/scheduling and facility location", *Transportation Sci.* 25 100-114

Loui, R. P., (1983), "Optimal Paths in Graphs with Stochastic or Multidimensional Weights", *Communications of the ACM,* 26, 670-676.

Martins, E.Q.V., (1984), "An Algorithm for Ranking Paths that May Contain Cycles", *European Journal of Operations Research*, 18, 123-130.

Miller-Hooks, E. D., (2001), "Adaptive Least-Expected Time Paths in Stochastic, Time-Varying Transportation and data networks", *Networks*, 37, 1, 35-52.

Miller-Hooks, E. D., and H. S. Mahmassani (2003), "Path comparisons for a priori and time-adaptive decisions in stochastic, time-varying networks", *European Journal of Operational Research,* Vol. 146, pp. 67-82.

Miller-Hooks, E. D., Mahmassani, H. S. (1998a), "Least possible time paths in stochastic, time-varying networks", *Compututations Operational Research*, 25, 1107–1125

Miller-Hooks, E. D., Mahmassani, H. S. (1998b), "Optimal routing of hazardous materials in stochastic, time-varying transportation networks", *Transportation Research Records*. 1645, 143–151.

Miller-Hooks, E. D., Mahmassani, H. S., (2000), "Least expected time paths in stochastic, time-varying transportation networks", *Transportation Science.* 34(2), 198-215.

Miller-Hooks, E., Mahmassani, H., (1998), "Least possible time paths in stochastic, time-varing networks," *Computers operations research*, 25, 12, 1107-1125.

Minieka, E. and Shier, D.R., (1973), "A note on an algebra for the k best routes in a network," *Journal of the society for Industral and Applied mathematics*, 145-149.

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

Moore, E.F. (1959), "The shortest path through a maze", *Proceedings of an International Synposium on the Theory of Switching (Cambridge, Massachusetts, 2-5 April, 1957),* Harvard University Press, Cambridge, 285-292.

Murthy, I., Sarkar, S., (1997), "Exact Algorithms for the Stochastic Shortest Path Problem with a decreasing deadline utility function", *European Journal of Operational Research*, 103, 209-229.

Nguyen, S., Pallottino, S., (1986). "Hyperpaths and Shortest Hyperpaths", *Combinatorial Optimization, Lecture Notes in Mathematics*, 1403, Springer-Verlag, Berlin, 258-271.

Nguyen, S., Pallottino, S., (1988), "Equilibrium Traffic Assignment for Large Scale Transit Networks", *European Journal of Operational Research*, 37, 176-186.

Nozick, L. K., List. G. F.,and Turnquist M. A., (1997), "Integrated routing and scheduling in hazardous materials transportation", *Transportation Science.* 31 200-215

Orda, A., Rom, (1990), "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-lengths", *Journal of the Association for Computing Machinery.* l. 37, 603-625.

Pallottino, S., (1984), "Shortest path methods: Complexity, interrelations and new propositions", *Networks,* 14, 257-267.

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

Perko, A., (1986), "Implementation of algorithms for K shortest loopless paths," *Networks*, 16, 149-160.

Polychronopoulos, G. H., Tsitsiklis, J. N., (1996), "Stochastic Shortest Path Problems with Recourse", *Networks*, 27, 133-143.

Powell, W. B., Jaillet, P., and Odoni, A. (1995), *Stochastic and Dynamic Networks and Routing*, Handbook in Operations Research and Management Science, Vol. 4, Networks, (M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, eds.), pp. 141-295, 1995.

Provan., J.S. (2003). "A polynomial-time algorithm to find shortest paths with recourse". *Networks*, 41(2):115–125,

Psaraftis, H. E., and Tsitsiklis, J. N., (1993), "Dynamic Shortest Path in acyclic networks with Markovian arc costs", *Operations Research*, 41, 1, 91-101.

Ross, S. M., (1993), *Introduction to Probability Models*, Academic Press, Inc. *Processes*

Sancho, N.G. F., (1988), "A New Type of Multi Objective Routing Problem," *Engineering Optimization*, 14, 115-119.

Schrank, D., and Lomax, R., (2005, May). The 2005 Urban Mobility Report, Texas Transportation Institute, The Texas A&M University System, http://mobility.tamu.edu

Scott, K. G., and D. Bernstein, (1998), "Solving a Best Path Problem when the Value of Time Function is Nonlinear", Presented at *the 77th Transportation Research Board Annual Meeting*, January 11-15, Washington, D.C.,

Sen, S., Pillai, R., Joshi, S., and Rathi, K., (2001) "A mean-variance model for route guidance in advanced traveler information systems", *Transportation Science*, 35, 1, 37-39.

Seok, J. and S. Pulat (2001), "On the stochastic shortest path problem and its promise as a project management tool", Master thesis.

Shier, D.R., (1976), "Iterative methods for determining the k shortest paths in a network," *Networks*, 6, 205-229.

Shier, D.R., (1979), "On algorithms for finding the k shortest paths in a network," *Networks*, 9, 195-214.

Sigal, L. E., Pritsker, A. A. B., and Solberg J. J., (1980), " The use of cutsets in Monte Carlo analysis of stochastic networks", *Mathematics Computer Simulation,* 21 376-384.

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

Spiess, H., Florian, M., (1989), "Optimal Strategies: A new Assignment Model for Transit Networks," *Transportation Research B* 23B, 83-102.

Stewart, B.S., and White, C.C., (1989), "Three Solution Procedures for Multi Objective Path Problems," *Control Theory and Advanced Technology*, 5, 4, 101-107.

Tarjan, R. E., (1983), *Data structures and Network Algorithms,* SIAM, Philadelphia, Pa.

Turnquist , M. A., (1987), "Routes, schedules, and risks in transportation of hazardous materials", B. Lev, J. A. Bloom, A. S. Gleit, F. H. Murphy, C. A. Shoemaker, eds. *Strategic Planning in Energy and Natural Resources*. North Holland, Amsterdam, The Netherlands, 289–302.

Warburton, A., (1987), "Approximation of Pareto Optima in Multiple-Objective, Shortest-Path Problems", *Operational Research*, 35 1, 70-79.

Wu, J., Florian, M., (1993), " A Simplical Decomposition Method for the Transit Equilibrium Assignment Problem", *Annals of Operations Research*, 44, 245-260.

Wu, J., Florian, M., and Marcotte, P., (1994), "Transit Equilibrium Assignment: A Model and Solution Algorithms", *Transportation Science* 28, 193-203.

Yen, J.Y., (1971), "Finding the K shortest loopless paths in a network," *Management Science*, 17, 712-716.

Yen, J.Y., (1971), "Finding the K shortest Loopless Paths in a Network", *Management Science*, 17, 711-715.

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