3-2003

Shortest Path Problems in a Stochastic and Dynamic Environment

Jae Il Cho

SHORTEST PATH PROBLEMS IN A STOCHASTIC AND DYNAMIC ENVIRONMENT

THESIS

Jae I. Cho, Captain, Republic of Korea Army

AFIT/GSE/ENS/03-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the United States Government.

AFIT/GSE/ENS/03-01

SHORTEST PATH PROBLEMS IN A STOCHASTIC AND DYNAMIC ENVIRONMENT

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Jae I. Cho, B.S.

Captain, Republic of Korea Army

March 2003

AFIT/GSE/ENS/03-01

SHORTEST PATH PROBLEMS IN A STOCHASTIC AND DYNAMIC ENVIRONMENT

Jae I. Cho, B.S.
Captain, Republic of Korea Army

Approved:

Dr. Jeffrey P. Kharoufeh Thesis Advisor	Date
Dr. James T. Moore Committee Member	Date

Acknowledgements

I would like to express my deepest appreciation to my thesis advisor Dr. Jeff Kharoufeh for giving me an valuable opportunity to study problem and for providing ardent guidance throughout my research. His passionate encouragement and profound expertise inspired me and enabled me to reach this state of accomplishment. Dr. James Moore also provided me very productive and constructive comments which made my research to be complete.

I also thank all of the world-class AFIT faculty members who raised a laymen to the level of this achievement and inspired me to infatuate with all valuable courses.

I am deeply indebted to my Korean Army for providing me this precious opportunity to pursue a master's degree with which I hope to better serve the further advancement of my Army.

Finally, I am deeply grateful for my parents who have heartened me to advance throughout my life and for being constantly concerned about my well-being in a foreign country. Also, I want to thank my fiancee for her persistent encouragement and love for me which supported me to delve into my studies. I would finally like to thank everyone around me here and in Korea for their help and support throughout my studies at AFIT.

Jae I. Cho

Table of Contents

List of Figures

List of Tables

AFIT/GSE/ENS/03-01

Abstract

In this research, we consider stochastic and dynamic transportation network problems. Particularly, we develop a variety of algorithms to solve the expected shortest path problem in addition to techniques for computing the total travel time distribution along a path in the network. First, we develop an algorithm for solving an independent expected shortest path problem. Next, we incorporate the inherent dependencies along successive links in two distinct ways to find the expected shortest path. Since the dependent expected shortest path problem cannot be solved with traditional deterministic approaches, we develop a heuristic based on the $K-$shortest path algorithm for this dependent stochastic network problem. Additionally, transient and asymptotic versions of the problem are considered. An algorithm to compute a parametric total travel time distribution for the shortest path is presented along with stochastically shortest path measures. The work extends the current literature on such problems by considering interactions on adjacent links.

SHORTEST PATH PROBLEMS IN A STOCHASTIC AND DYNAMIC ENVIRONMENT

1. Introduction

1.1 Background

Deterministic network theory has produced immense applications and contributions not only in mathematical solution techniques, but also in many technological areas ranging from communications systems to modern internet infrastructure. Particularly, finding the shortest path in a network has allured researchers and practitioners in a wide variety of applications involving the transport of materials from an origin to a destination.

In this thesis, we deal with the shortest path problem in a transportation network. Our approach to the transportation network problem can be applied in several other network areas such as telecommunication systems or manufacturing systems. For the transportation network, the deterministic shortest path problem still has great significance; however, it does not truly reflect the probabilistic nature of the varying cost of using arcs in a network. Intrinsically, the cost of an arc in the network of a deterministic shortest path problem is assumed to be static and constant, regardless of the environment in which the real network resides. However, in the real world, the cost of an arc in a network can hardly be deterministic or constant. For example, the required time to traverse an arc is not constant. Thus, one of the most commonly used ways of incorporating the varying aspect of cost in a network is a probabilistic approach to the cost.

One of the earliest researchers to impose the probabilistic property on arcs was Frank [1] who introduced a means to compute the shortest path cumulative distribu-

tion function. Since then, a great deal of stochastic shortest path problem research has been done. Particularly, stochastic network applications have recently become very widely used in computer science especially in modern internet infrastructure development. Despite variations in how to capture the uncertainty of link weight in a network, all models have in common some sort of probability associated with link weight. This probabilistic association is an improvement over the deterministic assumption in that it provides a way to capture the uncertain nature of the real world network.

In this thesis, we focus specifically on the transportation network problem and the associated stochastic network where each link is governed by some probabilistic measure. In our work, we use the terminology *link* instead of the standard *arc* when we focus on our primary subject, transportation networks as opposed to general networks.

Numerous efforts have been made to incorporate the reality of network uncertainty in stochastic models. One approach is the naive fitted model approach for each varying link weight based on observed data in the real world. Another is to capture the stochastic link weight evolution over some time period. Adding the dynamic nature of the network increases the reality of the network model; however, the analysis becomes more complicated. In most dynamic and stochastic network models, it is assumed that the probability distribution of all link weights is known *a priori*. This demands arduous effort to collect observed data.

To incorporate the probabilistic link weight behavior with a minimum data collection effort, many researchers utilize stochastic process models where the link weight is governed by a stochastic process model. Some stochastic models used to capture the traffic environmental variation affecting the link travel time are provided in [11] and [20].

One recent contribution due to Kharoufeh and Gautam [25] considered the modelling of the link travel time distribution using continuous time stochastic pro-

cesses to govern the velocity of a vehicle traversing a link. This work uniquely incorporated the environmental effect on the velocity of the traversing vehicle. Further, the environment process approach enables one to compute the distribution of random travel times explicitly. However, this work has yet to be extended to an entire network problem for the purpose of finding the stochastic and dynamic shortest path across the network with a given origin node and destination node. In this thesis, we extend the environment process approach developed by Kharoufeh and Gautam [25] to construct a stochastic network and to find the shortest path across this network using the environment process model.

1.2 Problem definition and Methodology

1.2.1 Problem statement

In this thesis, we extend the environment process model developed by Kharoufeh and Gautam [25] and consider an entire network. Further, we incorporate real-world effects such as traffic intensity and other conditions. Next, we study the inherent dependency between successive links in a path of the stochastic network and develop various algorithms to find the expected shortest path from the source node to the destination node. Finally, we examine the travel time distribution along a path for the sake of completeness and consider the stochastically shortest path problem.

1.2.2 Research Objectives

The objectives of this research may be summarized as follows. First, we enhance the characteristics of the environment process model to build the stochastic network model in accordance with real-world conditions. Next, we develop an algorithm to find the expected shortest path from source node to destination node assuming independence along links in the path. Then we propose various approaches to impose dependence between successive links in the path by considering real-world

dependence and develop an algorithm to find this dependent expected shortest path in a stochastic network. Finally, we develop an algorithm to find a total travel time distribution along a path. Once a path travel time distribution is found, we use it to find the stochastically shortest path in a network using various stochastically optimal shortest path measures.

1.2.3 Research Methodology

This research requires knowledge of the theory of continuous-time Markov chains (CTMC), Laplace transforms and deterministic network shortest path theory. To build the stochastic network model using the environment process approach, we will use CTMC analysis to model the effects of the environment on the travel time. We use the traditional deterministic shortest path procedures to build the independent expected shortest path algorithm in a stochastic network. We analyze the behavioral aspects of the environment process to incorporate the dependency between successive links in a path.

Once the dependency has been imposed, we develop a heuristic algorithm to find the dependent expected shortest path using a deterministic K-shortest path algorithm. Finally, convolution of individual link travel time along a path is studied, and we build the path travel time distribution extending the parametric result of Kharoufeh and Gautam [24]. Moreover, with approximated parametric distribution of the path travel time, we develop a stochastically shortest path optimality concept.

1.2.4 Contribution

This research explicitly extends the work of Kharoufeh and Gautam [25],[24] to model a stochastic network and constructs unique ways of finding the expected shortest path in the stochastic network based on individual environment process properties. Further, by imposing the dependency of a real traffic situation with two approaches, this study presents a very efficient and unique heuristic to find the

dependent expected shortest path. Finally, in this thesis, we introduce a method of finding the total travel time distribution of a path in a stochastic network that models on environment process. No such approaches exist in the operations research literature to date.

1.3 Outline of the thesis

The next chapter reviews the literature relevant to stochastic shortest path problems. In Chapter 3, the formal mathematical development of the models is presented. Chapter 4 presents numerical experimentation and implementation using the models of Chapter 3. In Chapter 5, we discuss the insights gained from the analysis of this problem, summarize the contributions of the work, and provide some directions for future research.

2. Review of the Literature

2.1 Introduction

The shortest path problem has been studied extensively in many fields including computer science, operations research, and transportation engineering. All stochastic shortest path works have the following in common: random arc length modelled as either a discrete or continuous random variable whose distribution is time-varying or time-invariant. Most of these works use a deterministic shortest path foundation to develop the algorithm to solve the stochastic shortest path problem. Thus, we first briefly review a standard deterministic shortest path algorithm along with some important deterministic network techniques used in our research. Then we examine the evolution of these approaches to a stochastic approach while primarily focusing on stochastic shortest path problem studies. Finally, we review the work of Kharoufeh and Gautam [25] which is the foundation of the dynamic and stochastic network modelling.

According to traditional deterministic shortest path studies, any deterministic shortest path algorithm can be categorized as either a label setting algorithm or label correcting algorithm depending on how the current solution (path) is iteratively improved. The most well known label setting shortest path algorithm is Dijkstra's algorithm. However, our problem is a dynamic and stochastic network shortest path problem. It has been accepted that the standard shortest path algorithms is applicable to compute the shortest paths in time-dependent (but not stochastic) networks [20]. But if the link weight is random and evolves over time, standard deterministic algorithms are not guaranteed to find the optimal path [11], [20]. In this context, for stochastic shortest path problem, we review the stochastic-static case and the stochastic-dynamic case separately. In the next section we review deterministic shortest path problems along with some of the network representation techniques used in our research.

2.2 Review of Deterministic Shortest Path Problems

A network algorithm's efficiency is heavily influenced by the network representation. The effective use of data structures can significantly improve the run times of an algorithm [9]. We introduce two of the network representation structures to be used in our research before reviewing Dijkstra's algorithm.

When we need to store the information about all arcs connected with a node, we use node-arc incidence matrix to store this information. For a graph $G(N, E)$ where N = node set and E=edge set, $n = |N|$ and $m = |E|$, this representation stores the networks in a $n \times m$ matrix which contains one row for each node of the network and one column for each arc as follows [9].

Node-Arc Incidence Matrix [9]: For an undirected graph $G(N, E)$, let V be an $n \times m$ matrix whose element $v_{ij} = 1$ if edge j is incident with node i and 0 otherwise.

In addition, we develop a new network representation to store the distance (arc length) between nodes in a undirected graph as follows.

Distance matrix: For an undirected graph $G(N, E)$, let D be an $n \times n$ matrix whose element $d_{ij} = c_{ij}$ if nodes i and j are connected with length c_{ij} and ∞ otherwise.

We use these two network representations to code algorithms in the Matlab environment.

Before we review a deterministic shortest path algorithm in detail, we review some relevant network definitions and the deterministic optimality condition of shortest path:

- A **cut** is a partition of the node set N into two parts, S and $\bar{S} = N - S$. Each cut defines a set of arcs consisting of those arcs that have one endpoint in S

and another endpoint in $\bar{S}$. Therefore, we refer to this set of arcs as a cut and represent it by the notation $[S, \bar{S}]$.

- A **tree** is a connected graph that contains no cycles.
- A tree T is a **spanning tree** of graph G if T spans all the nodes N of G.

The optimality condition of shortest path in a deterministic network is as follows,

Optimality condition [12]: Suppose a graph $G(N, E)$. Then, for every node $j \in N$, let $d(j)$ denote the length of some directed path from the source node to node j. Then the numbers $d(j)$ represent shortest path distances if and only if they satisfy the following shortest path optimality conditions:

$$d(j) \leq d(i) + c_{ij} \text{ for all } (i, j) \in A.$$

This condition is used in our study, particularly in developing the stochastic expected shortest path algorithm. Next, we briefly describe Dijkstra's algorithm which utilizes the above optimality condition iteratively to attain the shortest path.

Dijkstra algorithm [12].

Let S = permanent node set. $\bar{S}$ = Temporary node set. $S \bigcup \bar{S} = N$. $d(i)$ = Distance label of node i. $P(i)$ = predecessor of node i .

Step 0 Initialization.

$S = \emptyset$; $\bar{S} = N$;

$d(i) = \infty$ for each node $i \in N$;

$d(s) = 0$; and $P(s) = 0$ where s = source node.

Step 1 Node selection.

let $i \in \bar{S}$ be a node for which $d(i) = min\{d(j) : j \in \bar{S}\}$.

Step 2 Distance Update.

$S := S \bigcup \{i\}$;

$\bar{S}$: $= \bar{S} - \{i\}$;

for each $(i, j) \in A(i)$ where $A(i)$ adjacent arc list to node i,

if $d(j) > d(i) + c_{ij}$ then $d(j) = d(i) + c_{ij}$ and $P(j) := i$.

If $|S| = n$, terminate otherwise goto Step 1.

Dijkstra's algorithm finds the shortest paths from the source node to all other nodes in a network with nonnegative arc lengths. This algorithm maintains a directed out-tree T rooted at the source that spans the nodes with finite distance labels [12]. This algorithm maintains every arc (i, j) to satisfy the optimality condition $d(j) = d(i) + c_{ij}$ with respect to current distance labels. Further, we need to note that at termination when distance labels represent shortest path distances, T is a shortest path tree [12].

There is an assumption in using Dijkstra's algorithm: a network must have non-negative arc lengths. All networks in our research satisfy this condition. Even though there have been lots of deterministic shortest path algorithms developed, this algorithm is fast and relatively simple to run. We will use this algorithm as a sub-algorithm to develop the expected shortest path algorithm in a stochastic network.

In the next section, we review relevant research pertaining to the stochastic shortest path problem, particularly the stochastic and static variety.

2.3 Stochastic and Static Shortest Path Problems

In this section, first we consider relevant literature dealing with stochastic path distribution. One of the earliest works on stochastic shortest path found the cumulative probability distribution of shortest path in a network whose each arc weight is assumed to be probabilistic [1]. In this work, the author presented how to estimate the cumulative distribution function (CDF) of the shortest path along which each arc has a random weight. This work is considered to be one of the foundations of stochastic shortest path in terms of the research of shortest path distribution in a

stochastic network. The author basically considered two cases to build a stochastic network: each arc having an unknown distribution with available observational data and each arc having a normal distribution. The former was a non-parametric analysis and the latter was a parametric analysis. Using statistical analysis and exponential transformation, he was able to construct the estimated CDF of the random length of the shortest path. However, while he introduced how to estimate the CDF of the possible shortest path in a network, he did not suggest how to find the shortest path. He only suggested the comparison of all path distributions in a network to find the shortest one. When each arc weight is assumed to be a normal random variable, he introduced how to compare two disjoint $s-t$ (source node to termination node) paths. This comparison method is relevant to our stochastic network model where each link travel time asymptotically follows a normal distribution [24] which is in Section 2.5. His comparison method is summarized below.

Let π_1 and π_2 be disjoint $s-t$ paths. In addition, let $|\pi_i|$ denote the length of path π_i. Then, the criterion to compare two paths is

$$P\{|\pi_1| \geq l_0\} < P\{|\pi_2| \geq l_0\}$$

where l_0 is a positive number. In other words, we could say the $|\pi_1|$ is l_0 better than $|\pi_2|$. Since all costs in the network are assumed to be normal random variables, obviously the sum of costs in the network is also normally distributed. Further, those arcs in a path are thought to be independent. If variances of two normal random $s-t$ path travel times are equal, $\sigma_1^2 = \sigma_2^2$, then

$$P\{|\pi_1| \geq l_0\} < P\{|\pi_2| \geq l_0\} \quad \text{iff} \quad \mu_1 < \mu_2.$$

If variances of two normal random $s-t$ path travel times are not equal, $\sigma_1^2 \neq \sigma_2^2$, then $P\{|\pi_1| > t\} \leq P\{|\pi_2| > t\}$ for t such that

$$\begin{cases} \text{if} \quad \sigma_1 > \sigma_2 \quad : \quad t \leq \frac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1} \\ \text{if} \quad \sigma_1 < \sigma_2 \quad : \quad t \geq \frac{\sigma_2\mu_1 - \sigma_1\mu_2}{\sigma_2 - \sigma_1} \end{cases}$$

This result is related to utilizing the path travel time distribution to find the stochastically shortest path in Section 3.5. Another relevant random variable distribution comparison concept is a stochastic ordering based on the following definition [15].

Definition A random variable X is said to be *stochastically less than* Y, written $X \leq_{st} Y$, if $P\{X > t\} \leq P\{Y > t\}$ for all $-\infty < t < \infty$.

Frank's research [1] informs us that if the path travel time is normally distributed, we cannot find the stochastically shortest path because of the distinct variance of individual arcs all across the stochastic network. However, the above two concepts help us to develop a stochastically shortest path measure in Chapter 3.

Many stochastic shortest path problems are related to stochastic PERT (Program Evaluation and Review Technique) networks. One early stochastic PERT network study to find the shortest path computed the probability distribution of project duration in a stochastic project network [7]. However, most of these early PERT studies assumed the task duration (arc) was a discrete random variable so that they can easily condition the successive arc duration with the preceding arc duration which would lead to the total duration distribution [7]. Since most task durations are expected to have a discrete value, this type of network makes sense, yet it does not provide an efficient technique in the continuous arc case. However, the discrete stochastic network approach enabled the authors to utilize discrete time Markov chain analysis techniques to compute the shortest path distribution [15]. In our research, each arc weight in a stochastic network is assumed to be a continuous random variable and is governed by a continuous time Markov chain model.

Next, Elliot, et al. [4] provided a formal performance measure for selecting a path of a stochastic network. In this work, the authors presented an analytic derivation of path optimality indices for directed acyclic networks. This concept contributed greatly to the evaluation of a path in a stochastic network. Prior to this work, most results focused on constructing a path length probability distribution [4]. In this work, the authors introduced an approach to account for the dependence among all paths sharing certain arcs and to find the optimality indices introduced by them in a stochastic network. This work is relevant to our study in terms of utilizing an estimated path travel time distribution of our stochastic network model. However, their research stopped short of a method to find the actual optimal path based on their path performance measure.

Harilos and Tsitskikls [11] introduced the *environment variable concept* which attempts to capture the environmental impacts on the random arc length. The environment variable concept was based on the motivation that the motion of a vehicle across a stochastic terrain is under the effect of favorable or unfavorable weather conditions. Their work was directly motivated by the routing of a ship across the ocean under uncertain and dynamically changing weather conditions. The network nodes represent geographical regions across which the ship will travel. Thus, this environment variable can be said to be a meteorological variable describing the status of condition affecting the random duration of sojourn. Their primary objective was to determine the vehicle's best action as a function of the state of the environment that prevails at a given node of the network. Following is their approach to incorporate the environment effect on random length establishing in order to find the shortest path. Define,

e_i, f_i**:** Environment variable at state $i \in S = 1, 2, ..., K$ and f_i is each corresponding cost.

X_m**:** The minimum achievable total expected cost of going across the arc given that the state of the environment is e_m, $m \in S$.

C**:** Waiting cost.

p_{mk}**:** Probability that the environment transitions from state m to k.

The objective function of the problem is,

$$X_m = min\left(f_m, C + \sum_{k=1}^{K} p_{mk} X_k\right), \quad m = 1, ..., K.$$

It is clear that there are two possible choices for the vehicle at any node: either depart immediately, and incur a cost of f_m, or wait, and incur an immediate waiting cost of C, and a residual expected cost of $\sum_{k=1}^{K} p_{mk} X_k$. When waiting until favorable environment state, the expected sum is, over each possible 'next' environment state e_k, the minimal expected cost associated with the vehicle facing a variable of state e_k after the transition and assuming an optimal policy thereafter. The environment variable evolvement is modelled as a DTMC with transition probability matrix $[p_{mk}]$ [11].

As solutions to find the shortest path according to above nonlinear equations, they presented three kinds of methods: successive approximations, policy iterations and parametric linear programming [11]. While this work has a unique contribution of incorporation of environment effect on random arc as a Markov process in a stochastic network, the cost of the arc cannot be modelled in continuous space and this approach intrinsically cannot capture the dependence between successive lengths which exist in real-world environment.

A very recent model extending the work of Harilos and Tsitsiklis [11] is the paper by Amir and Farhad [26]. They assumed the environmental variable evolves in accordance with independent semi-Markov processes instead of Markov processes. Further, they assume the length of each arc is an exponential random variable whose parameter is a function of the environmental variable of its initiative node instead of deterministic function. So the transitions of each environmental variable influence the parameters of the exponential distributions of the lengths of the related emanating arcs and, consequently, the expected lengths of the outgoing arcs. In this work

they consider two cases to find the optimal strategy at each node; when arriving at a node, either the state of environmental variables of adjacent nodes or all nodes in a network are assumed to be known. The authors developed algorithms for optimal strategy of movement in each node of acyclic networks in terms of expected arrival time from source to sink node using semi-Markov decision process techniques and linear programming. However, their major drawback is the exponential explosion of the complexity of the relevant algorithms when they consider the case where each node's environmental variable state is assumed to be known upon arrival to any node. This work successfully generalized the work by Harilos and Tsitsklis [11]. However, this approach tends to be very complex as the size of problem grows and it still fails to capture the dependency of adjacent arcs in a stochastic network which exists in the real world.

In the next section, we review research pertaining to stochastic shortest path problems in a dynamic environment which is directly related to our problem.

2.4 Stochastic and Dynamic Shortest Path Problems

Several works address the problem of determining the shortest path in stochastic, yet static networks. By static, we mean that the time-invariance property of a random arc in any kind of stochastic network. Many works address the determination of the probability distribution, or expected value, of minimum path length in stochastic, static networks including the efforts of finding the paths with maximum utility value [22]. We need to first clarify the meaning of dynamic before discussing this class of problem. According to *Stochastic and Dynamic Networks and Routing* (Ball,T.L et al [16]), a problem is *dynamic* if one or more of its parameters is a function of time. One of the breakthrough works to incorporate the time-varying reality of arc length is the work by Miller-Hooks and Mahmassani [21]. At first, they sought the least possible time paths in stochastic, time-varying networks based on the perception of non-FIFO (First in First out) networks. "In non-FIFO networks,

the travel time on a path may be shorter if one arrives at an intermediate node later than it would be if one arrived earlier". This is mainly due to the time-varying property of random arcs in a network. Their major contributions are the development of algorithms to find the least possible time paths in a network with stochastic, time-varying arc weights, including computation of the probabilities. In terms of explicit consideration of the time-varying aspect of a random arc, their work has great significance. Following are the main contributions of their work.

Miller-Hooks and Mahmassani [21] consider a scenario in which the travel time distribution function along an arc does not change over all possible time horizons, rather, it evolves over time during certain intervals (e.g. peak period: $t_0 \leq t \leq t_0 + I\delta$, stationary period: $t > t_0 + I\delta$). Further, they discretiezed the time unit during peak period i.e.)$t \in \{t_0 + n\delta\}$, where $n = 0, 1, ..I$. Their algorithm can be summarized as follows: Define,

Graph $G(\nu, \mathcal{A}, \mathcal{L}, \mathcal{T}, \mathcal{P})$**:** ν=finite set of nodes. $\mathcal{A}$= a set of arcs. $\mathcal{L}$=a set of discrete times $\{t_0 + n\delta\}, n = 1, 2..., I$. $\mathcal{T}$=nonnegative real valued possible travel times. $\mathcal{P}$= a set of probabilities associated with travel times,$\mathcal{T}$.

$\tau_{i,j}^k(t)$**:** Traversing time from node i to j at time t where $k = 1, ..., K_{i,j}(t)$.

$K_{i,j}(t)$**:** The number of possible distinct travel time values on arc (i, j) at time t.

$\rho_{i,j}^k(t)$**:** The probability associated with the occurrence of $\tau_{i,j}^k(t)$.

$\lambda_j(t)$**:** The upper bound on the minimum travel time from node j at time t.

$\eta_i(t)$**:** The temporary label of $\lambda_i(t)$.

The methodology recursively updates the least possible travel time from the current node to destination node with the following objective function.

$$\eta_i(t) = min_p\{\tau_{i,j}^p + \lambda_j(t + \tau_{i,j}^p(t))\}$$

where p is the set of indices of possible travel times on arc (i, j) at time t. The main concept in this work is to update the temporary label of an arc using a deterministic label correcting algorithm [12] iteratively until all nodes are permanently updated (least possible time from source node to sink node found).

After this work, the least expected time path research was done by the same authors [23]. In a manner similar to the above approach, they develop a label correcting algorithm to find the least expected shortest time path in a so-called stochastic time-varying network. Yet, this research also limited the time-varying range of a random arc distribution function in a stochastic network to certain peak periods and beyond it, it was assumed to be stationary. While this limited range of time-varying random arc distribution function incorporated the time-varying reality of the world, it does not sufficiently cover all possible time intervals. In summary, the authors' work established a significant step to incorporate the time-varying reality although it does not sufficiently incorporate the whole range of the time-varying domain.

A dynamic, stochastic and dependent stochastic shortest path approach was addressed in a paper by Fu and Rillet [20]. Their work is unique in terms of the dependence they imposed in their stochastic network. This paper is greatly relevant to our study for several reasons which will become apparent. We briefly review their work next. The main contributions of the paper is the extension of the shortest path problem in dynamic and stochastic networks (DSSPP) where link travel times evolve according to a continuous-time Markov chain (CTMC). Moreover, it explicitly incorporates the dependency between successive arcs on a path which is very useful and significant to our study. In addition, they demonstrated that the traditional deterministic shortest path algorithms may not guarantee the optimal shortest path in a DSSPP. This failure of traditional deterministic shortest path algorithms mainly is due to the dependency of successive arcs in DSSPP. Next we present a brief description of their main works [20].

Consider a directed graph in which each link weight is represented by a random travel time with an associated probability distribution. Link travel time distribution is dependent on the time of day (i.e the time a link is entered). The travel time on these links is modelled as a CTMC. Further, travel time on each link is not independent which is realistic and is a significant factor in developing the expected travel time approximation.

Let $\{X_a(t), t \in T\}$ be a stochastic process where $X_a(t)$ is the travel time for vehicles entering link a at time t, and T is a continuous parameter set $T = [0, \infty)$. Assume T to be time range limited. Then there exists probability density function, $f_{X_a}(x_a, t)$ at any possible time range t. Let $\mu_{X_a}(t)$ be the mean of the stochastic process $\{X_a(t), t \in T\}$ as follows:

$$\mu_{X_a}(t) = E[X_a(t)] = \int_0^\infty x_a f_{x_a}(x_a, t) dx_a.$$

Similarly the variance of the individual random variable $X_a(t)$ is;

$$\nu_{X_a}(t) = E[(X_a(t) - \mu_{X_a}(t))^2] = \int_0^\infty (x_a - \mu_{X_a}(t))^2 f_{X_a}(x_a, t) dx_a.$$

The problem is to find the expected shortest path p^* from origin node to destination node. This problem is referred to as a *dynamic and stochastic shortest path problem* (DSSPP). Let W_p denote the total travel time on path p and $f_{W_p}(W_p)$ denote the PDF of W_p where $p \in P$, and P is set of paths. Then the formal problem statement is

$$\textbf{(DSSPP)}\ p^* = argmin_{p \in P} E[W_p].$$

The main concept of this work is first to let random variable Y_i denote the arrival time at node i. Y_i is assumed to be equal to the departure time at node i or the time link a is entered. Let Z_a be the random travel time on link a. Then, the final random route travel time to node j from source node is given by the following recursive expression:

$$Y_j = Y_i + Z_a.$$

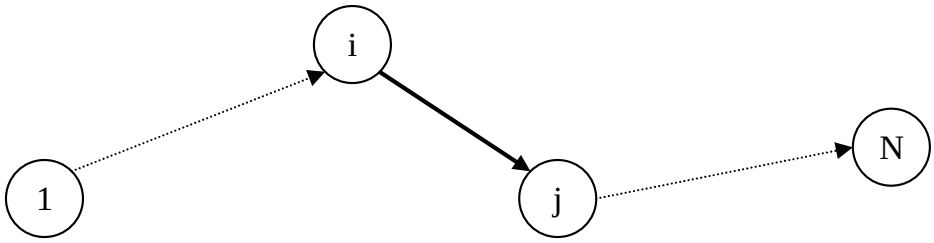

Figure 2.1 A path p from origin node 1 to destination node N [20].

The mean route travel time is found as follows,

$$\begin{aligned} E[Y_j] &= E[Y_i] + E[Z_a] \\ &= E[Y_i] + E\left[E[Z_a|Y_i]\right] \\ &= E[Y_i] + E[\mu_{x_a}(Y_i)]. \end{aligned}$$

Using the Taylor series approximation of the function $\mu_{x_a}(t)$ above at $t = E[Y_i]$, they establish approximation formulas for the above equations and found out that in successive links in a DSSPP, there exists explicit dependence coming from the variance of the preceding link travel time. Further, due to this dependency, they showed that traditional deterministic algorithms may not be applicable to find the optimal shortest path. Consequently, they provide a heuristic to find the optimal shortest path in a DSSPP.

The expected travel time on individual link, μ_{X_a} varies drastically depending on the preceding path chosen. Consequently, they capture the dependence between successive links in a DSSPP in their model and develop the algorithm to solve it. Finally, imposing the dependence of links along a path in DSSPP has great relevance to our research.

2.5 Stochastic Model for Individual Links

In this section, we review the stochastic model for individual travel times [24], [25]. This stochastic model for travel time on an individual link is called *the environment process methodology.* Kharoufeh and Gautam [25] derived an analytical expression for the cumulative distribution function (CDF) of travel time on a stochastic transportation link. This approach explicitly captures the effect of environmental factors (e.g., roadway geometry, traffic density, weather condition, etc.) on the velocity of a vehicle traversing the link. These possible environmental factors are melted into a random environment stochastic process controlling the velocity of vehicle. Figure 2.2 gives a dual relationship between the random time to traverse a link of length x, $T(x)$ and the cumulative distance travelled up to time t, $D(t)$. In this figure, the slope of each chord corresponds to the speed of the vehicle which

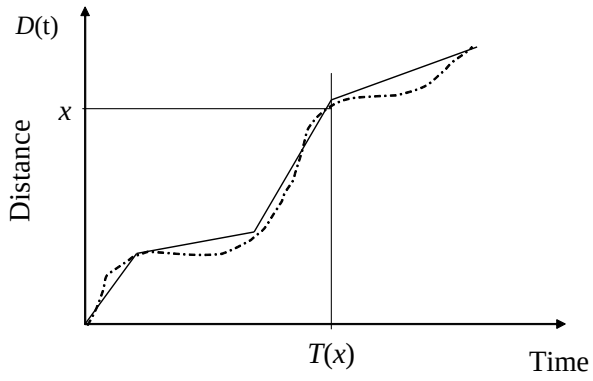

Figure 2.2 Estimated trajectory of vehicle using piece-wise, linear functions [25].

is governed by the stochastic environment process over continuous time. We can induce the following the dual relationship between $T(x)$ and $D(t)$ which allows one to find the distribution of $T(x)$ by deriving the distribution of $D(t)$.

Lemma 2.1 *Event* $\{D(t) \leq x\} \equiv$ *Event* $\{T(x) \geq t\}$ $\forall x,\ t \geq 0$.

This relationship is obvious from Figure 2.2. The environment process is modelled as a continuous time Markov chain (CTMC) governing the velocity of vehicle

traversing the link. A random environment is thought to evolve over continuous time staying in one state of the finite dimensional state space before making a transition to a different state. Each environment state corresponds to a unique velocity of the vehicle through any mapping function from environment state to velocity of the vehicle.

Let $\{Z(t) : t \geq 0\}$ denote the random environment process modelled as a CTMC and $\nu = \{V_1, V_2, V_3, ..., V_K\}$ denote the possible velocity set of a vehicle. We can partition actual continuous velocity values into a discretized velocity range or value. This simplification is necessary to maintain one-to-one mapping between the finite environment state and the corresponding velocity of the vehicle. Figure 2.3 depicts this mapping between the environment process and the velocity of the vehicle. Consider a link having length, $x > 0$. The velocity of a vehicle at time

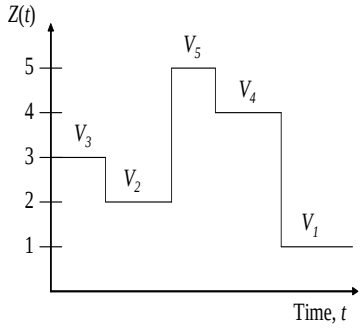

Figure 2.3 A sample path of the environment process,$\{Z(t) : t \geq 0\}$.

t is governed by a random environment process, $\{Z(t) : t \geq 0\}$. The finite state space of the environment process, $S = \{1, 2, ..., K\}$ maps to finite velocity set, $\nu = \{V_1, V_2, V_3, ..., V_K\}$. The mapping function $f : S \rightarrow \nu$ describes how the speed varies as the environment process evolves. Defining this function can be specific to the particular scenario. The probability distribution of travel time for a link of length x is defined as,

$$G(x; t) \equiv P\{T(x) \leq t\} \tag{2.1}$$

Using Lemma 1, it is seen that

$$G(x;t) \equiv P\{T(x) \leq t\} = 1 - P\{D(t) \leq x\}, \quad x, t \geq 0. \tag{2.2}$$

Now define the following joint probability distribution,

$$H_i(x,t) = P\{D(t) \leq x, Z(t) = i\}, i \in S \tag{2.3}$$

where $H_i(x,t)$ is the joint probability that the vehicle has travelled a distance no more than x and the environment process is in state $i \in S$ at the time. The procedure to find the eventual travel time distribution is to find the joint distribution $H_i(x,t)$ and apply it in Equation (2.2) to find the travel time distribution. This procedure can be summarized as

$$\begin{aligned} G(x;t) \equiv P\{T(x) \leq t\} &= 1 - P\{D(t) \leq x\} & (2.4) \\ &= 1 - \sum_{i \in S} H_i(x,t). & (2.5) \end{aligned}$$

The main results of this model, which form the basis of our methodology, are given in Theorem 2.1.

Theorem 2.1 *(Kharoufeh and Gautam [25]) If the random environment process* $\{Z(t) : t \geq 0\}$ *governing vehicle speed is a continuous-time Markov chain with infinitesimal generator,* $Q = [q_{i,j}]$*, then* $H_i(x,t)$ *satisfies the partial differential equation*

$$\frac{\partial H_i(x,t)}{\partial t} + \frac{\partial H_i(x,t)}{\partial x} V_i = \sum_{j \in S} q_{ji} H_i(x,t), \ i \in S \tag{2.6}$$

with initial condition

$$H_i(x,0) = B_i(x) = P\{Z(0) = i\}.$$

The differential equation is not easily solved so the Laplace transform technique was employed to obtain a solution. Following are results of the two dimensional transform over x and t.

$$\tilde{H}^*(s_1, s_2) = \tilde{B}(s_1)(s_1 V + s_2 I - Q)^{-1} \tag{2.7}$$

where $H^*(x, s_2) = [H_i^*(x, s_2)]_{i \in S}$ is the row vector of the Laplace transform of $H^*(x, t)$ with respect to t, $\tilde{B}(s_1)$ is a $1 \times K$ row vector, s_1 and s_2 are complex transform variables with $Re(s_1) > 0, Re(s_2) > 0$ and $V \equiv diag(V_1, V_2, ..., V_K)$. This result is obtained through twice Laplace transformation where s_2 corresponds to the Laplace transformation variable with respect to t and s_1 is the Laplace-Stieltjes transformation variable with respect to x.

Finally, in order to obtain the distribution of travel time, we need to invert the above matrix and sum over all possible $i \in S$ states. This requires us to use two dimensional numerical inversion.

$$\begin{aligned} P\{T(x) \leq t\} &= 1 - P\{D(t) \leq x\} \\ &= 1 - \mathcal{L}^{-1} \left\{ \sum_{i \in S} \frac{1}{s_1} \tilde{H}_i^*(s_1, s_2) \right\} \end{aligned}$$

where $\mathcal{L}^{-1}$ denotes the inverse Laplace operator.

For the purposes of this research, it is necessary to describe the three required elements in the model for link travel time distribution:

Velocity Function: The velocity function maps a particular environment state, $i \in S$, to a corresponding velocity, V_i. Hence, the velocity function is a function of environment state, namely $f(i)$. It can be of any form as long as it maintains a one-to-one mapping from S to ν.

Q Matrix: The infinitesimal generator matrix, Q, incorporates the rate of transition from state i to j. In a practical sense, those rates can be estimated statistically from observations of the vehicle's speed transitions [25].

Initial State Distribution: Initial distribution vector, z_0, describes the initial environment state a vehicle undergo. This can also be estimated through observed data of the initial speed of vehicles. The function $f(i)$ allows us to directly estimate the initial state distribution from initial velocity data.

Next we review the analysis of link travel time moments for this stochastic environment process model which is used to develop the algorithm of expected shortest path in Chapter 3. First we present the transient analysis result and then the asymptotic analysis.

2.5.1 Transient analysis

We need to analyze this model in depth to obtain important measures such as moments and steady state distribution. We review the moments of random travel time by utilizing the Laplace transform property. In general, the r^{th} moment of the Laplace transform function is obtained by

$$m_r(x) \equiv E[(T(x))^r] = (-1)^r \frac{d^r}{ds^r} \tilde{G}(s)\Big|_{s=0.} \tag{2.8}$$

where $\tilde{G}(s)$ is the Laplace-Steiltjes transform of $G(.)$, the CDF of x.

By the above relationship, the general expression for the r_{th} moment of the conditional travel time was given as [24].

$$\tilde{K}_0^r(s_1) = r! z_0 (s_1 V - Q)^{-r} e. \tag{2.9}$$

where z_0 is the initial distribution vector. This equation gives an exact analytical expression for the LST of the r_{th} moment of the random travel time, provided that

all derivatives exist at $s_2 = 0$. The actual numeric value of the moment can be obtained though one-dimensional numerical Laplace inversion. In most cases, we are interested in 1^{st} and 2^{nd} moments by which we can find the mean travel time and variance. Hence, the mean travel time and variance of travel time on a link are as follows,

Mean travel time, $m_1(x)$:

$$m_1(x) = \mathcal{L}^{-1}[s_1^{-1} z_0 (s_1 V - Q)^{-1} e], \tag{2.10}$$

the second moment of travel time, $m_2(x)$:

$$m_2(x) = \mathcal{L}^{-1}\left[2 s_1^{-1} z_0 (s_1 V - Q)^{-1} (s_1 V - Q)^{-1} e\right], \tag{2.11}$$

the variance of travel time, $v(x)$:

$$v(x) = m_2(x) - m_1(x)^2. \tag{2.12}$$

It is important to note that these are direct results where the initial distribution vector (z_0) influences the moments of travel time. The analysis of a CTMC in which the initial distribution is considered is called a transient analysis compared to an asymptotic analysis where CTMC behavior is independent of the initial distribution, we call the above measures transient mean and variance, respetively. This transient analysis explicitly considers the impact of initial environment distribution on the travel time. However, as the travel length grows, one expects the impact of the initial environment state to diminish. This environment process which is a CTMC will have a steady-state distribution provided the infinitesimal generator matrix Q is ergodic. In a practical sense, as the travel length grows, surely the impact of the initial speed of a vehicle on the travel time should dwindle as long as the vehicle is

supposed to change its speed often enough. In the next subsection, we review the asymptotic results of Kharoufeh and Gautam [25].

2.5.2 Asymptotic analysis

According to basic CTMC analysis, if an embedded DTMC is irreducible and positive recurrent, then it is called ergodic for which there always exists a steady-state distribution. That means as long as all non-diagonal elements in the infinitesimal generator Q matrix are non-zero values, this CTMC system is ergodic and this can be easily seen by following Figure 2.4. According to the fundamental CTMC analysis

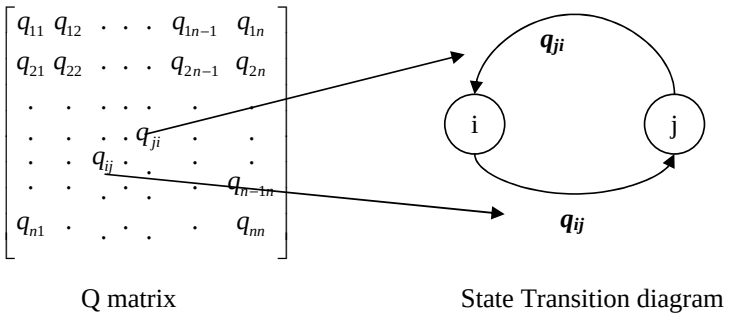

Figure 2.4 Relationship between non-zero q_{ij} and recurrence of transition.

[15], let $p = [p_j]$ be the steady state distribution for $j \in S$ where S is the sample space of the CTMC. Then,

$$\begin{aligned} pQ &= 0 \\ \sum_{j \in S} p_j &= 1. \end{aligned}$$

Using the above result, we can easily derive the steady state distribution of environment process $\{Z(t) : t \geq 0\}$. We combine the above two equations into one matrix equation for computational purposes. Since there are n unknowns and $n+1$

equations, one of the $n+1$ equations is redundant. Hence,

$$[p_k]_{1\times n} \cdot [\tilde{Q}_{i,j}]_{n\times n-1} = [\tilde{e}]_{n\times 1}$$

where $[\tilde{Q}_{ij}] =$

$$\begin{bmatrix} -q_1 & q_{12} & q_{1n-1} & 1 \\ q_{21} & -q_2 & q_{2n-1} & 1 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ q_{n1} & q_{n2} & q_{nn-1} & 1 \end{bmatrix}$$

and $[\tilde{e}] =$

$$\begin{bmatrix} 0 \\ . \\ . \\ . \\ 1 \end{bmatrix}$$

By replacing the last column of the Q matrix with a unit column $[1, 1, ..., 1]$, we make one matrix equation as above. Next, we look into the asymptotic mean of $T(x)$.

According to formal proof (Kharoufeh and Gautam [25]), as $x \longrightarrow \infty$,

$$\frac{m_1(x)}{x} \longrightarrow \frac{1}{pv}. \tag{2.13}$$

where v is the velocity vector i.e.) $v = Ve$. The second moment of asymptotic travel time was derived in a similar manner, so as $x \longrightarrow \infty$, it can be shown (Kharoufeh and Gautam [25])

$$\frac{m_2(x)}{x^2} \longrightarrow \frac{1}{(pv)^2}.$$

The above results, along with Chebyshev's inequality prove [24] that , as $x \to \infty$,

$$\frac{T(x)}{x} \longrightarrow \frac{1}{pv}.$$

This is intuitive result that as $x \to \infty$ the travel time asymptotically converges to x/pv. However, we still need to capture the behavior of the variance of travel time as the length of the link tends toward infinity in order to estimate the asymptotic distribution and asymptotic variation. Hence, Laplace-stieltjes transform approach to asymptotic variance is used and shows that the standard deviation of travel time $(T(x))$ in the asymptotic region is proportional to the square root of the link length (Kharoufeh and Gautam [24]).

Let $\eta_i, i = 1, 2, \ldots, K$, denote the K eigenvalues of $V^{-1}Q$, and let l_i (r_i), $i = 1, 2, \ldots, K$, denote their corresponding left (right) eigenvectors. Of the K eigenvalues, one eigenvalue is zero and the remaining $K - 1$ are strictly negative. In particular, we note that the vector $\hat{p}$ is the left eigenvector corresponding to the zero eigenvalue.

$$\lim_{x\to\infty} \frac{Var[T(x)]}{x} = -\frac{2}{pv} \sum_{i=2}^{K} \frac{1}{\eta_i} \frac{(pr_i)(l_i V^{-1} e)}{l_i r_i}.$$

For detailed proof, refer to Kharoufeh and Gautam [24]. Therefore, for large x the asymptotic mean and variance can be obtained as follows,

Asymptotic mean travel time, $m_1(x)$:

$$m_1(x) = \frac{x}{pv} \tag{2.14}$$

Asymptotic second moment of travel time, $m_2(x)$:

$$m_2(x) = \left(\frac{x}{pv}\right)^2 \tag{2.15}$$

Asymptotic variance of travel time,$v(x)$:

$$v(x) = -\frac{2x}{pv}\sum_{i=2}^{K}\frac{1}{\eta_i}\frac{(pr_i)(l_i V^{-1} e)}{l_i r_i} \tag{2.16}$$

Since computational effort and time to obtain the numerical inversion of travel time distribution can be excessive and the inversion process itself is at times unstable, we need to approximate the distribution parametrically in order to extend to the network problem where the CDF of each link is needed to compute shortest paths. According to Kharoufeh and Gautam's [25] work, to solve a 10-state problem, it requires about 8.5×10^{11} floating point operations in Matlab for two-dimensional inversion.

By using a surrogate distribution, we can utilize approximated parametric properties to compute the total path travel time distribution with computational feasibility. Since we have already developed the formulas for computing moments of the conditional travel time distribution (transient and asymptotic), we can parameterize each individual link travel time distribution with the three required elements mentioned earlier; Q matrix, initial distribution, and velocity function $f(i)$. Then we can use far less computational effort to compute the total travel time distribution utilizing the parameterized distribution.

In this chapter, we reviewed several important contributions regarding the stochastic shortest path problem and individual link travel time methodology. The objective of this work is to extend the main work of Kharoufeh and Gautam [25], [24] to an entire stochastic and dynamic network rather than a single link. The ultimate objective is to develop a general framework for a stochastic and dynamic shortest path algorithm in which the arc costs are random and dynamic travel times.

3. Formal Mathematical Model

In this chapter, we present the formal extension of the environment process methodology for a stochastic and dynamic network. This involves the models for the static and dynamic expected shortest path as well as models for the stochastically shortest path.

3.1 Extension to stochastic network model

There are three factors required for the stochastic environment process model: initial environment state distribution, generator matrix (Q) and the velocity function. With these three elements, we attempt to capture the realistic and intrinsic traffic characteristics into our stochastic and dynamic network model. Each of the three factors is presented in detail next.

3.1.1 The generator matrix

The elements of the generator matrix represent the transition rate from environment state i to j in unit time. Practically, this can obtained from observed data of the speed change rate (Kharoufeh and Gautam [25]). In our study, we generate realistic representations of the Q matrix reflecting various traffic types which exist in the real world. Suppose a frequently congested traffic environment exists on certain links. Then one can imagine that the more frequent velocity changes and higher variance of velocity would exist on this link. On the other hand, on a fairly stable link such as interstate freeway, a vehicle would undergo fewer velocity transitions and its velocity range would be limited. These interpretations of traffic phenomena give us a logical idea about how to construct the Q matrix according to various types of traffic conditions.

We need to recall that the off-diagonal elements in the Q matrix, q_{ij} correspond to the transition rate from state i to j which can be directly interpreted as velocity

transition from V_i to V_j accordingly. These ideas lead us to set the general Q matrix definitions.

High density traffic link: Large number of environment state spaces, wide range of q_{ij}. Example-Arterial road networks, computer networks.

Moderate density traffic link: Medium number of environment state spaces, medium range of q_{ij}. Example-Suburban roadways.

Low density traffic link: Small number of environment state spaces, narrow range of q_{ij}. Example-Interstate freeway segment.

These descriptions are in accordance with traffic theory. In the next chapter, we generate the Q matrix of each link based on the network topology.

3.1.2 Velocity function

Our approach implicitly captures the time-dependency of velocity and the environmental effect which can also reflect the number of vehicles on road. However, there are certain requirements for this velocity function. It should maintain a one-to-one mapping and be a function of the environment state. As long as this functional requirement is met, the velocity function may assume many forms. In our study, we consider the following two kinds of velocity functions possible.

For arbitrary constant C,

Linear model:

$$V = \frac{C}{i} \quad \forall \quad i \in S$$

Exponential model:

$$V = \frac{C}{e^i} \quad \forall \quad i \in S$$

where S is the sample space of CTMC $\{Z(t) : t \geq 0\}$. For example, for five environment state space case such that $S = \{1, 2, ..., 5\}$ with $C = 25$ (miles/hour), the

mapped linear velocity space $V = \{\frac{25}{1}, \frac{25}{2}, .., \frac{25}{5}\}$. In both models, as the environment state (i) value increases, the velocity decreases. As we can foresee the different effects of each of the above models, the exponential model would cause more dramatic velocity differences between varying environment states which eventually reinforce the effect of transition rate on the travel time on a link. We examine in greater detail this sensitivity. The above models are computationally simplistic and incorporate the varying velocity as it depends on the environment state.

3.1.3 Initial distribution

The initial state of the environment, $\{Z(0) = i\}$, implies that at time 0, the velocity is V_i. In an individual link, one can legitimately assume that initial velocity is zero or at least the lowest velocity (V_1). We need to recall the environment state definition in terms of a vehicle traversing a link. The environment state is a reflection of the condition that a vehicle undergoes on a link at a specific time $Z(t)$. Further, the environment process of a link is initiated at the time a vehicle enters the link (more precisely, the starting node). Thus, the time t of environment state $Z(t)$ explicitly is the time elapsed from the departure of a vehicle from the starting node of a link. Suppose a vehicle travels two consecutive links in a network $(0, 1)$, $(1, 2)$ as in Figure 3.1. It started from node 0 at $T = 0$ where T is the random travel time on the path. Then the travel time T at certain time t after passing the intermediate node is the summation of two time segments t_1, t_2. As noted above, since there are two distinct CTMC: $\{Z_{01}(t)\}, \{Z_{12}(t)\}$, there exist two distinct random travel times on the path $0 \rightarrow 1 \rightarrow 2$: t_1, t_2.

When considering dependence of successive random link travel times t_1, t_2, the initial state $Z(0)$ of following links in a path might not be the lowest state and this is studied in Section 3.3.

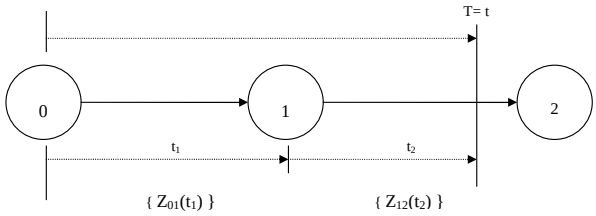

Figure 3.1 Dynamic in a network.

3.1.4 Stochastic and dynamic network model

With the variation of these three factors (Q matrix, velocity function and initial distribution) in environment process CTMC depending on each distinct link's characteristics, we are able to model an environment stochastic network. In Figure 3.2, we show how each link can have unique variations of CTMC governing its travel time.

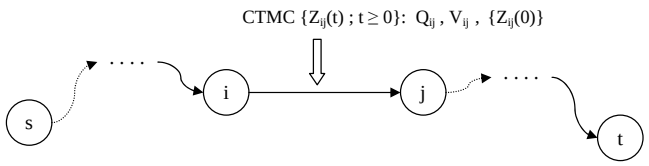

Figure 3.2 An arbitrary path shortest path in a stochastic network.

In the above s-t path, each link possesses a distinct environment process CTMC.

Next, based on these modelling approaches, we study the expected shortest path in a stochastic network and examine the dependency between successive travel time links.

3.2 Independent Expected Shortest Path Methodology

In this section, we present an algorithm to find the optimal expected shortest path when each link in the network is assumed to be independent. In other words,

on a given path, each link's travel time T_{ij} is independent so that preceding link travel history does not affect the subsequent link travel time. Thus, each link has its own environment process; initial state distribution $Z(0)$, generator matrix Q and velocity function $f(\cdot)$. Since our objective is to find the expected shortest path in a given network, once we find the mean travel time of all links, we can treat this problem as a traditional deterministic shortest path problem where all arc weights are given as the mean static travel time. We use both the transient and asymptotic mean travel time in finding the expected shortest path in a network for comparison purposes. The algorithm for this problem consists of 3 stages as follows:

Stage 0: Initialization.

Let $X = [x_{ij}]$ be distance matrix, where x_{ij}= distance along link (i, j).

Let Q_{ij}, V_{ij} be generator matrix (Q) and velocity function matrix, respectively, for each link (i, j)

Stage 1: Finding the transient and asymptotic mean travel time of all links in a network. Matrix $A = [a_{ij}]$ where a_{ij} is the mean travel time of link (i, j).

Transient mean travel time: $a_{ij} = \mathcal{L}^{-1}\left[s_1^{-1} z_0 (s_1 V_{ij} - Q_{ij})^{-1} e\right]$

Asymptotic mean travel time: $a_{ij} = m_1(x_{ij}) = \dfrac{x_{ij}}{pv}$.

Stage 2: Finding the shortest path using a traditional algorithm based on mean travel time of each link.

To find the shortest path based on the mean travel time of each link, we use the traditional Dijkstra's algorithm [12]. Dijkstra's algorithm is a well-known algorithm to find the deterministic shortest path between a given pair of nodes using the following optimality condition iteratively to form the shortest spanning tree.

$$d(j) \leq d(i) + c_{ij}, \quad \forall \quad (i, j) \in A$$

where $d(j)$ denotes the distance to node j from the source node. In this approach, we use the mean travel time as the deterministic constant length of each link (i, j) and apply the Dijkstra's algorithm to find the expected shortest travel time path.

Next we study the more realistic problem involving link dependencies.

3.3 Dependent Expected Shortest Path Methodology

Thus far, we have assumed that link travel times are mutually independent. In this section, we relax this assumption and capture the dependency of downstream links on a path.

In the environment process model, the environment state $\{Z(t) : t \geq 0\}$ is a CTMC which evolves over continuous time starting with initial distribution, $Z_0 = [P\{Z(0) = i\}]_{i \in S}$. The state $Z(t)$ at any time t determines the unique velocity value of a vehicle traversing the link. So, the initial state distribution determines the initial velocity of the vehicle. In the independent network case, we arbitrarily assumed that the initial distribution of the environment is $P\{Z(0) = 1\} = 1$ and $P\{Z(0) = i; \forall i \neq 1\} = 0$. In our nondecreasing velocity model (either linear or exponential) with increasing state value, this means that initial velocity of the vehicle is the lowest velocity at time 0.

However, with this initial distribution approach, distinct paths in a network contain no history of previous links travelled. In real-world context, the history of the previous links the vehicle travelled affects the following link travel time. In a continuous travel on a path, the initial velocity on a successive link varies according to the preceding link. For this reason, we must incorporate this dependency in finding the expected shortest path in a network. Yet, there are several ways to impose this dependency into our model. For example, we can incorporate this dependency by considering the expected environment state at the completion of one sojourn of a

link. Another approach is to consider the probability distribution of the environment state at the end of the sojourn.

3.3.1 Method of Expected Terminal distribution

This approach is based on the following arguments. All definitions are in reference to Figure 3.3:
Define,

$T(x_{ij})$:= Random time to travel along link (i, j).

t_{ij} := A realization of the random variable $T(x_{ij})$.

$\{Z_{ij}(t) : t \geq 0\}$:= The environment process governing link (i, j).

Q_{ij} := Generator matrix for link (i, j).

S_{ij} := Sample space for $\{Z_{ij}(t) : t \geq 0\}$.

Assumption : $S_{ij} = S_{jk} = S$ where S= the sample space of the CTMC.

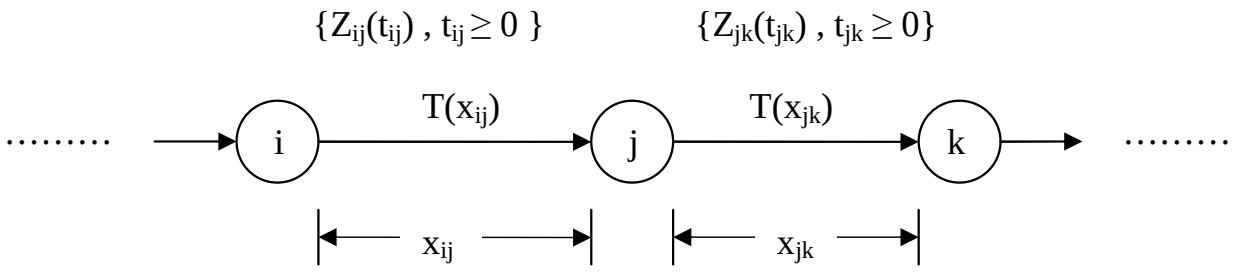

Figure 3.3 Expected probability of preceding link.

In Figure 3.3, our objective is to impose a downstream dependency on link travel time $T(x_{jk})$ on the given path via the initial distribution of the environment state $\{Z_{jk}(0)\}$. First, it is important to note that an additional assumption for this approach is the sample spaces of the adjacent arcs are identical. In other words, $S_{ij} = S_{jk}$ holds so that the distribution of the environment state of the preceding

link at the time the vehicle enters the following link becomes the initial distribution of the environment state of the next link. Then suppose at time t_{ij} the vehicle enters the next node through link (i, j) to begin its travel on the following link, (j, k). Each link has its own environment process $\{Z_{ij}(t_{ij})\}, \{Z_{jk}(t_{jk})\}$, respectively. Then our dependency approach is described as follows,

$$\begin{aligned} P\{Z_{jk}(0) = m\} &= P\{Z_{ij}(T(x_{ij})) = m\} && (3.1) \\ &= \sum_{h \in S_{ij}} P\{Z_{ij}(T(x_{ij})) = m | Z_{ij}(0) = h\} P\{Z_{ij}(0) = h\} && (3.2) \end{aligned}$$

where $m \in S_{jk}, S_{ij}$. Thus, for the initial state distribution vector $[P\{Z_{jk}(0)\}]$,

$$\begin{aligned} [P\{Z_{jk}(0)\}]_{S_{jk}} &= [P\{Z_{ij}(0)\}]_{S_{ij}} \cdot [P\{Z_{ij}(t_{ij}) = h | Z_{ij}(0) = g\}]_{S_{ij}} \\ &= [P\{Z_{ij}(0)\}]_{S_{ij}} \cdot [P_{gh}\{Z_{ij}(t_{ij})\}]_{S_{ij}} && (3.3) \end{aligned}$$

where $h, k \in S_{ij}$ and $P_{gh}\{Z_{ij}(t_{ij})\}$ is the transition probability from state g to h for time t_{ij} and g, $h \in S_{ij}$.

The realization of $T(x_{ij})$, t_{ij} is the time the vehicle enters the node j after traversing link (i, j). In other words, we impose the explicit dependency of the following link environment process on the preceding environment process evolvement at entering time t_{ij}. Thus, we only need to determine the time t_{ij} of entering the following link to impose dependency.

As a first approach, we decide to take the expected probability distribution of environment state from the preceding arc as the initial distribution of the following link, namely

$$[P\{Z_{jk}(0)\}]_{S_{jk}} = [E[P\{Z_{ij}(T(x_{ij}))\}]]_{S_{ij}}$$

This method is defined as the expected terminal distribution approach because of the dependence on the expected terminal environment process distribution of the preceding link (i, j).

Finally our method of expected terminal distribution is stated as follows. The initial probability of the environment state m upon random arrival time $T(x_{ij})$ at node j is given by,

$$[P\{Z_{jk}(0)\}]_{S_{jk}} = [E[P\{Z_{ij}(T(x_{ij}))\}]]_{S_{ij}} \tag{3.4}$$

However, finding $[E[P\{Z_{ij}(T(x_{ij}))\}]]_{S_{ij}}$ is not trivial. For a state $h \in S_{ij}$, $E\left[P\{Z_{ij}(T(x_{ij})) = h\}\right]$ is clearly a function of random travel time $T(x_{ij})$. Thus, this is given as follows,

$$E\left[P\{Z_{ij}(T(x_{ij})) = h\}\right] = \int_0^\infty P\{Z_{ij}(t_{ij}) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij} \tag{3.5}$$

Obviously Equation (3.5) can only be applied when the probability density function (PDF) of random travel time, $f_{T(x_{ij})}(t_{ij})$, is available. There is not an explicit closed PDF form available. So, $f_{T(x_{ij})}(t_{ij})$ must be estimated by either parametric methods or empirical methods (Section 3.5). In addition, somehow we have to determine the t_{ij}, the realization of $T(x_{ij})$ when a vehicle completes its sojourn on link (i, j).

So, we can either use an approximate technique to find the Equation (3.5) or arbitrarily specify the time t_{ij} such as $t_{ij} = E[T(x_{ij})]$. One approximate technique for finding the function $P\{Z_{ij}(t_{ij})\}$ would be a Talylor expansion at $t_{ij} = E[T(x_{ij})]$ as follows,

$$\begin{aligned} P\{Z_{ij}(t_{ij}) = h\} &= P\left\{Z_{ij}(E[T(x_{ij})]) = h\right\} + P'\left\{Z_{ij}(E[T(x_{ij})]) = h\right\}(t_{ij} - E[T(x_{ij})]) + \\ &\quad \frac{1}{2}P''\left\{Z_{ij}(E[T(x_{ij})]) = h\right\}(t_{ij} - E[T(x_{ij})])^2 + \cdots \end{aligned} \tag{3.6}$$

provided that $P\{Z_{ij}(t_{ij}) = h\}$ is infinitely differentiable at $t_{ij} = E[T(x_{ij})]$. If we truncate at the linear term by assuming the second and higher order derivatives are

negligible, we obtain the first order approximation,

$$P\{Z_{ij}(t_{ij}) = h\} \simeq P\{Z_{ij}(E[T(x_{ij})]) = h\} + P'\{Z_{ij}(E[T(x_{ij})]) = h\} \cdot (t_{ij} - E[T(x_{ij})]) \quad (3.7)$$

Then we substitute into Equation (3.5) as follows:

$$\begin{aligned}
\int_0^\infty P\{Z_{ij}(t_{ij}) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij} &\simeq \int_0^\infty \left\{P\{Z_{ij}(E[T(x_{ij})]) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij}\right\} + \\
&\quad \int_0^\infty \left\{t_{ij} P'\{Z_{ij}(E[T(x_{ij})]) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij}\right\} \\
&\quad - \int_0^\infty E[T(x_{ij})] P'\{Z_{ij}(E[T(x_{ij})]) = h\} \\
&\quad \cdot f_{T(x_{ij})}(t_{ij}) dt_{ij} \\
&= P\{Z_{ij}(E[T(x_{ij})]) = h\} \int_0^\infty \left\{f_{T(x_{ij})}(t_{ij}) dt_{ij}\right\} + \\
&\quad P'\{Z_{ij}(E[T(x_{ij})]) = h\} \int_0^\infty \left\{t_{ij} f_{T(x_{ij})}(t_{ij}) dt_{ij}\right\} \\
&\quad - E[T(x_{ij})] P'\{Z_{ij}(E[T(x_{ij})]) = h\} \\
&\quad \cdot \int_0^\infty \left\{f_{T(x_{ij})}(t_{ij}) dt_{ij}\right\} \\
&= P\{Z_{ij}(E[T(x_{ij})]) = h\} + \\
&\quad P'\{Z_{ij}(E[T(x_{ij})]) = h\} (E[T(x_{ij})] - E[T(x_{ij})]) \\
&= P\{Z_{ij}(E[T(x_{ij})]) = h\} + 0 \quad (3.8)
\end{aligned}$$

Hence, with the first order approximation of $P\{Z_{ij}(t_{ij}) = h\}$, we are able to approximate Equation (3.5)

Proposition 3.1 *The approximated initial environment state distribution of link* (j,k), $[P\{Z_{jk}(0)\}]_{S_{jk}}$ *is given as,*

$$[P\{Z_{jk}(0)\}]_{S_{jk}} = [E[P\{Z_{ij}(T(x_{ij}))\}]]_{S_{ij}} \simeq [P\{Z_{ij}(m_1(x_{ij}))\}]_{S_{ij}} \quad (3.9)$$

where $m_1(x_{ij})$=$E[T(x_{ij})]$ *provided that* $S_{jk} = S_{ij}$.

Proof. This is a direct result of Equations (3.5), (3.8), (3.4). Q.E.D.

By using the above result, we incorporate the dependency of the successive downstream link travel time. In this approach, we implicitly reflect the preceding path history into the successive travel time through the initial distribution. In order to compute the $[P\{Z_{ij}(t_{ij})\}]_{S_{ij}}$, we need the initial distribution of CTMC $\{Z_{ij}(T(x_{ij})) : T(x_{ij}) \geq 0\}$, $[P\{Z_{ij}(0) = h\}]_{h \in S_{ij}}$, and this is also obtained from its preceding expected travel time probability and so on. Thus, the following link travel time, $m_1(t_{jk})$ explicitly depends on the preceding link environment process probability vector, $[P\{Z_{ij}(t_{ij})\}]_{S_{ij}}$.

Suppose there is a path η in a network as shown in Figure 3.4. Then we can make a node set $N(\eta)$= the set of all nodes on path η. Further, if we arrange the node set $N(\eta)$ in an ascending order number of nodes such that $N(\eta) = \{0, 1, \cdot\cdot j \cdot\cdot, k-1, k\}$ where 0 denotes the source node, k denotes the sink node and j is the jth node in $N(\eta)$. Let Y_j be the total travel time up to node $j \in N(\eta)$ and let $T_{j-1,j}$ be the

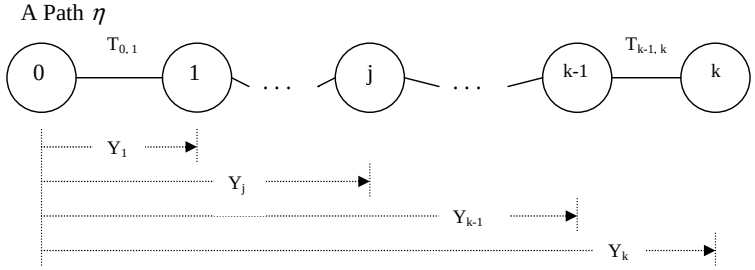

Figure 3.4 A sample path η in a stochastic network.

random travel time from the $(j-1)st$ node to the jth node. Then, our objective is to find the total dependent travel time Y_k. For Y_k, Equation (3.10) holds.

$$Y_k = Y_{k-1} + T_{k-1,k} \tag{3.10}$$

In other words, the sum of the total travel time up to node $k-1$ and the last link travel time, $T_{k-1,k}$, is the total travel time to sink node k, Y_k. Since we seek the expected total travel time, $E[Y_k]$,

$$\begin{aligned} E[Y_k] &= E[Y_{k-1}] + E[T_{k-1,k}] && (3.11) \\ &= E[Y_{k-1}] + E\left[E[T_{k-1,k}|T_{k-2,k-1}]\right] && (3.12) \end{aligned}$$

Namely, due to the dependence of successive links in the path from node 0 to k, we can condition upon the preceding link random travel time $T_{k-2,k-1}$ to find the expected travel time of $T_{k-1,k}$.

Since the random travel time of the next link on a path only depends on the current link travel time via the expected probability method, this concept is similar to the Markovian property where transition occurs in the same manner. We can find the dependent expected travel time $E[Y_k]$ by using the above formula recursively as follows.

$$\begin{aligned} E[Y_k] &= E[Y_{k-1}] + E[T_{k-1,k}] \\ &= E[Y_{k-2} + T_{k-2,k-1}] + E\left[E[T_{k-1,k}|T_{k-2,k-1} = t_{k-2,k-1}]\right] \\ &= E[T_{0,1}] + E\left[E[T_{1,2}|T_{0,1} = t_{0,1}]\right] + \cdots + E\left[E[T_{k-1,k}|T_{k-2,k-1} = t_{k-2,k-1}]\right] \\ &= E[T_{0,1}] + \mu_{T_{1,2}}(t_{0,1}) + \cdots + \mu_{T_{k-1,k}}(t_{k-2,k-1}) && (3.13) \\ &= E[T_{0,1}] + \sum_{j=1}^{k-1} \mu_{T_{j,j+1}}(t_{j-1,j}) && (3.14) \end{aligned}$$

where $\mu_{T_{j,j+1}}(t_{j-1,j}) = E\left[E[T_{j,j+1}|T_{j-1,j} = t_{j-1,j}]\right]$, the expected travel time on link $(j, j+1)$ given the travel time of the preceding link $(j-1, j)$ is $t_{j-1,j}$.

Hence, the total travel time sums up expected total travel time at the preceding node, $E[Y_j]$ and the expected travel time on the current link, $\mu_{T_{j,j+1}}(t_{j-1,j})$. We can compute the total expected travel time $E[Y_k]$ recursively using Equation (3.14).

Next, in order to compute the dependent expected travel time $\mu_{T_{j,j+1}}(t_{j-1,j})$, we need to define the Laplace-Stieltjes transform of the initial distribution of the next link $(j, j+1)$.

In Chapter 2, we introduced the joint probability distribution $H_i(x,t) = P\{D(t) \leq x, Z(t) = i\}, i \in S$. Now, the initial joint probability distribution in a dependent network is given as follows provided $H(x_{j,j+1}, t_{j,j+1}) = [H_i(x_{j,j+1}, t_{j,j+1})]_{i \in S_{j,j+1}}$,

$$\begin{aligned} H_i(x_{j,j+1}, 0) &= P\{D(0) \leq x_{j,j+1}, Z_{j,j+1}(0) = i\} \\ &= P\{Z_{j,j+1}(0) = i\} \end{aligned}$$

At time $t_{j-1,j}$ (the time a vehicle enters link $(j, j+1)$), the random travel time of link $(j, j+1)$, $t_{j,j+1}$ is initiated. Further, using Equation (3.9),

$$P\{Z_{j,j+1}(0) = i\} = P\{Z_{j-1,j}(E[t_{j-1,j}]) = i\} = H_i(x_{j,j+1}, E[t_{j-1,j}])$$

Taking the Laplace-Stieltjes transform of $H(x_{j,j+1}, E[t_{j-1,j}])$ with respect to $x_{j,j+1}$,

$$\begin{aligned} LST(H(x_{j,j+1}, E[t_{j-1,j}])) &= \int_0^\infty e^{s_1 x_{j,j+1}} dH(x_{j,j+1}, E[t_{j-1,j}]) \\ &= \tilde{B}(s_1, E[t_{j-1,j}]) \end{aligned}$$

where $H(x_{j,j+1}, E[t_{j-1,j}])$ is the vector of $H_i(x_{j,j+1}, E[t_{j-1,j}])$. Now $\mu_{T_{j,j+1}}(t_{j-1,j})$ is computed as follows,

$$\begin{aligned} \mu_{T_{j,j+1}}(t_{j-1,j}) &= E\left[E[T_{j,j+1} | T_{j-1,j} = t_{j-1,j}]\right] \\ &= \mathcal{L}^{-1}\left[s_1^{-1} \tilde{B}_{j,j+1}(s_1, E[t_{j-1,j}])(s_1 V - Q_{j,j+1})^{-1} e\right] \end{aligned} \quad (3.15)$$

Equation (3.15) enables us to compute the conditional expected travel time of following links successively. Each index $(j, j+1)$ refers to the $(j+1)th$ link in the path $N(\eta)$.

The above computation procedures are seemingly tedious; however, practically the above procedures only differ from independent method (section 3.1) in terms of the variation of the initial distribution $[\tilde{B}]$ in Equation (2.10).

Finally we summarize the algorithm for computing the expected total travel time on a path with terminal distribution dependency on successive links.

Approach I: Expected terminal distribution.

Step 1 Compute the expected travel time $E[T_{ij}]$ on link $x_{ij} \Rightarrow m_1(x_{ij})$

Step 2 Compute the expected environment state probability vector of link (i, j), $[P\{Z_{ij}(m_1(x_{ij})\}]$

Step 3 Equate the initial environment state probability vector of link x_{jk} with the expected probability vector found in step 2. $\Rightarrow [P\{Z_{jk}(0)\}] = [P\{Z_{ij}(m_1(x_{ij})\}]$

Step 4 Compute the expected travel time on link x_{jk} with Equation (3.15). $\Rightarrow$ $\mu_{T_{jk}}(t_{ij}) = \mathcal{L}^{-1}\left[s_1^{-1}\tilde{B}_{ij}(s_1, E[t_{ij}])(s_1 V - Q)^{-1}e\right]$.

Step 5 Compute the total expected travel time by Equation (3.14). $\Rightarrow E[Y_k] = E[T_{0,1}] + \sum_{j=1}^{k-1} \mu_{T_{j,j+1}}(t_{j-1,j})$.

Next, we present another dependence model which may possibly more accurately reflect the dependency of network travel times.

3.3.2 Method of Expected Terminal State

A vehicle traversing the link x_{jk} in Figure 3.3 initially starts the travel with a specific initial environment state, i. That is $P\{Z_{jk}(0) = i\}_{i \in S_2} = 1$. In the previous approach, we imposed the dependency of the next link travel time through the expected initial state probability vector of link x_{jk} provided that the vehicle arrives from link x_{ij}. In this section, we focus more on a specific initial state the vehicle should begin with rather than the probability distribution of the initial state. In other words, we seek the expected state of the initial distribution and not the

distribution itself. Thus, we want to find the $E[Z_{jk}(0)]$ and we assume that a vehicle begins its travel on link x_{jk} at state $E[Z_{jk}(0)]$ with probability 1.

However, in general, this expected terminal state is not an integer value corresponding to one of the states of the following link (j,k). In order to maintain the matching of this expected terminal state to the initial state by the following link's CTMC, $\{Z_{jk}(t_{jk}) : t_{jk} \geq 0\}$, we utilize a function $nint[u]$ which denotes the nearest integer to u.

In this section, we seek $E[Z_{ij}(T(x_{ij}))]$, the expected terminal state given by,

$$\begin{aligned} E[Z_{ij}(T(x_{ij}))] &= \sum_{h \in S_{ij}} \{Z_{ij}(T(x_{ij})) = h\} P\{Z_{ij}(T(x_{ij})) = h\} \\ &= \sum_{h \in S_{ij}} h P\{Z_{ij}(T(x_{ij})) = h\} \end{aligned} \tag{3.16}$$

As shown above, $E[Z_{ij}(T(x_{ij}))]$ is a function of $T(x_{ij})$ and via conditioning upon $T(x_{ij})$ it can be represented as,

$$\begin{aligned} E[Z_{ij}(T(x_{ij}))] &= \sum_{h \in S_{ij}} h P\{Z_{ij}(T(x_{ij})) = h\} \\ &= \sum_{h \in S_{ij}} h \int_0^\infty P\{Z_{ij}(T(x_{ij})) = h | T(x_{ij}) = t_{ij}\} f_{T(x_{ij})}(t_{ij}) dt_{ij} \\ &= \sum_{h \in S_{ij}} h \int_0^\infty P\{Z_{ij}(t_{ij}) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij} \end{aligned} \tag{3.17}$$

where $f_{T(x_{ij})}(\cdot)$ is the PDF of t_{ij} which is the time of finishing the travel on link (i,j).

As in the previous section, we can approximate the above probability function $P\{Z_{ij}(t_{ij}) = h\}$ with a second order Taylor series expansion (Equations (3.7), (3.8))

as follows,

$$E[Z_{ij}(T(x_{ij}))] = \sum_{h \in S_{ij}} h \int_0^{\infty} P\{Z_{ij}(t_{ij}) = h\} f_{T(x_{ij})}(t_{ij}) dt_{ij} \quad (3.18)$$

$$\simeq \sum_{h \in S_{ij}} h P\{Z_{ij}(E[T(x_{ij})]) = h\} \quad (3.19)$$

Finally, the following proposition holds,

Proposition 3.2 *The first order approximated expected terminal state of preceding link* (i, j) *at* $t_{ij} = E[T(x_{ij})]$ *is given as,*

$$E[Z_{ij}(T(x_{ij}))] \simeq nint \left[\sum_{h \in S_{ij}} h P\{Z_{ij}(m_1(x_{ij})) = h\} \right] \quad (3.20)$$

where $m_1(x_{ij}) = E[T(x_{ij})]$ *and nint[.] is the nearest integer function.*

Proof. This is a direct result of Equations (3.17) and (3.19). Q.E.D.

Since the expected value of a state may not be an integer value, we round off to make an integer value matching the environment state. Proposition 3.2 states that the following link (j, k)'s initial environment state is the state of preceding link (i, j) at the expected travel time of link (i, j). So the initial state probability vector of the following link x_{jk} is,

$$[P\{Z_{jk}(0) = 1\}, P\{Z_{jk}(0) = 2\}, ., P\{Z_{jk}(0) = i\}, ...P\{Z_{jk}(0) = K\}]$$
$$= [0, 0, ., P\{Z_{jk}(0) = i\}, ...0]$$

where $P\{Z_{jk}(0) = i\} = 1$, $i = E[Z_{jk}(0)]$. This dependency has a dramatic impact on the following link travel time depending on the preceding link as we will show in Chapter 4. Instead of imposing the dependency of the probability distribution of

the initial state of the following link, this approach explicitly seeks the dependency of the expected initial state.

Finally this approach is summarized as follows.

Approach II: Expected Terminal State.

Step 1 Compute the expected travel time $E[T_{ij}]$ on link (i,j).

Step 2 Compute the environment state probability vector of link (i,j), $[P\{Z_{ij}(m_1(x_{ij})\}]$.

Step 3 Compute the rounded expected state,

$$E[Z_{ij}(t_{ij})] \Rightarrow nint\left[\sum_{h\in S_{ij}} hP\{Z_{ij}(m_1(x_{ij})) = h\}\right].$$

Step 4 Compute the expected travel time on link x_{jk} with Equation (3.15)$\Rightarrow \mu_{T_{jk}}(t_{ij}) = \mathcal{L}^{-1}\left[s_1^{-1}\tilde{B}_{ij}(s_1, E[t_{ij}])(s_1V_{jk} - Q_{jk})^{-1}e\right]$.

Step 5 Compute the total expected travel time by Equation (3.14) $\Rightarrow E[Y_k] = E[T_{0,1}] + \sum_{j=1}^{k-1} \mu_{T_{j,j+1}}(t_{j-1,j})$.

Next, we consider another dependence approach which would require less computational effort while imposing the same kind of dependence used thus far.

3.3.3 Method of Asymptotic dependence

Thus far we have imposed the dependence of the following link's CTMC on the preceding terminal state and the expected terminal state probability distribution. Since the time a vehicle enters a downstream link is the key to impose the dependency between successive links, we consider the asymptotic behavior of the subsequent link. Due to the limiting behavior of an ergodic CTMC, this approach is computationally much more efficient. The key concept of this approach is imposing the dependency via the asymptotic behavior of CTMC.

First, for the method of expected terminal distribution, consider two successive links on a path (Figure 3.3), (i,j) and (j,k). The distribution of the environment at

the start of the sojourn of link (j,k) is given by,

$$\begin{aligned}[P\{Z_{jk}(0)=h\}]_{h\in S_{jk}} &= \lim_{t\to\infty}\left[E[P\{Z_{ij}(t)=h\}]_{h\in S_{ij}}\right] \\ &= \left[E[\pi^h_{i,j}]_{h\in S_{ij}}\right] \\ &= \pi^h_{i,j} \end{aligned} \tag{3.21}$$

where $\pi^h_{ij} = \lim_{t\to\infty} P\{Z_{ij}(t)=h\}$ is the steady-state distribution of CTMC of link (i,j) and provided $S_{ij} = S_{jk}$. We omit both sample space notations; S_{ij}, S_{jk} for convenience henceforth.

Next, for the method of expected terminal state, we can use the following Equation (3.22).

$$\lim_{t\to\infty} E[Z_{ij}(t)] \simeq nint\left[\sum_{h\in S_{ij}} h\pi^h_{ij}\right] \tag{3.22}$$

Next is the link's CTMC initial distribution for $m \in S_{jk}$,

$$P\{Z_{jk}(0)=m\} = 1 \text{ where } m = nint\left[\sum_{h\in S_{ij}} hP_{Z_{ij}}(h)\right]$$

With this asymptotic approach, we can still use both methods developed in earlier sections as shown above: the method of expected terminal distribution and method of expected terminal state. The only difference is to use the asymptotic probability vector of the preceding link $[\pi^h_{ij}]$ instead of the transient probability vector $[P\{Z_{ij}(t_{ij})\}]$. In fact, using the asymptotic measures would considerably reduce computational time because imposing the dependence between successive links instead of computing the mean travel time of the preceding link (i,j) in computing the transient state probability $[P\{Z_{ij}(E(t_{ij}))\}]$ requires less effort. We can easily compute the steady state probability π^h_{ij} using the Equations (3.23) and (3.24),

$$[\pi^h_{ij}][Q_{ij}] = 0 \tag{3.23}$$

$$[\pi^h_{ij}]e = 1 \tag{3.24}$$

This method differs from the two previous methods in terms of the termination time of the preceding link's CTMC ($T(x_{ij})$). Computationally, this method is much easier and further, if the CTMC is well behaved (converging quickly to steady state), then this approach would be much different from previous approaches.

Next, we study algorithms for the expected dependent shortest path utilizing two dependency approaches.

3.4 Algorithms for the Expected Dependent Shortest Path

In contrast to the independent network case, it is now noted that the classical deterministic algorithm may not find the expected dependent shortest path due to the dependency between two consecutive link travel times. In fact, the standard deterministic shortest path algorithm may not guarantee the shortest path as shown in following proposition.

Proposition 3.3 *If a successive individual link travel time in a network depends on a preceding link travel time, the standard shortest path algorithm may fail to find the expected shortest path between two nodes.*

It is a well-known fact that a greedy algorithm such as Dijkstra's algorithm may fail to find the shortest path in a dependent network [20]. Thus, we present two algorithms to solve this dependent expected shortest path problem along with a linear programming approach.

3.4.1 Explicit Enumeration

One naive approach would be to compute expected travel times for all possible paths from the source to the sink. This would be computationally inefficient as the number of nodes grows. However, an explicit enumeration algorithm would give the exact solution of the dependent expected shortest path.

Let φ be the set of all distinct paths from the source node to the sink node. Further, let p be a path in set P. That is, $P = \{p : p \text{ is a path from the source to sink node}\}$.

All path enumeration algorithm

Step 1 Enumerate all paths p (trees) emanating from the source node.

Step 2 Compute the total travel time of each p. For each link $(i, j) \in p$ successively find the expected total travel time $E[Y_p]$ using Equation (3.14).

Step 3 Rank all $E[Y_p]$ according to each value and choose the minimum $E[Y_{p^*}]$ Thus, p^* is the optimal expected shortest path.

These procedures require intensive computational effort not only to compute each path's travel time, but also to find all possible paths.

In the next subsection, we present a similar approach using linear programming method to show how even a simple dependent stochastic network can be intractable.

3.4.2 Linear Programming

In general, the linear programming approach to the deterministic shortest path problem tends to be computationally slower than a network algorithm [12]. However, in this section we present a general linear programming approach to the shortest path problem in a stochastic network to complete the research work.

At first, for a general dependent stochastic network $G(N, E)$ where N is the node set and E is the link set of graph G, define linear programming variables as follows,

s:= Source node and t:= Sink node.

$\widetilde{\psi}$:= The independent links set in graph $G(N, E)$. The independent links are always the links emanating from source node s (Initial links).

ψ:= The dependent links set in graph $G(N, E)$. Further, $\psi \cup \widetilde{\psi} = E$ and $\psi \cap \widetilde{\psi} = \emptyset$.

$A(j,k)$:= The set of preceding nodes to link (j,k).

$c_{(i,j)}$:= The mean random travel time to traverse the independent link (i,j) for all $(i,j) \in \widetilde{\psi}$.

$c_{i,(j,k)}$:= The mean random travel time to traverse the dependent link (j,k) from node i for all $(j,k) \in \psi$.

$X_{i,(j,k)}$:= 1 if the link (j,k) (dependent link) from node i is in the shortest path, otherwise, 0.

$X_{(i,j)}$:= 1 if the link (i,j) (independent link) is in the shortest path, otherwise, 0.

Then, the general linear integer programming formulation of expected shortest path in a dependent stochastic network is stated as follows,

$$\text{Minimize} \quad \sum_{(s,j)\in\widetilde{\psi}} c_{(s,j)} X_{(s,j)} + \sum_{(j,k)\in\psi} \sum_{i\in A(j,k)} c_{i,(j,k)} X_{i,(j,k)}$$

Subject to

$$\sum_{(j,k)\in\widetilde{\psi}} X_{(j,k)} = 1 \text{ (For source node } j = s)$$

$$\sum_{i\in A(k,j)} \sum_{(k,j)\in\psi} X_{i,(j,k)} - \sum_{(j,k)\in\psi} \sum_{i\in A(j,k)} X_{i,(j,k)} = 0$$

(For all node $j \in N - \{s,t\}$)

$$\sum_{(k,j)\in\psi} \sum_{i\in A(k,j)} X_{i,(k,j)} = 1 \text{ (For sink node } j = t)$$

$$\forall \quad X_{i,(j,k)}, X_{(j,k)} \in \{0,1\}$$

In our stochastic dependent network, depending on the preceding link, the mean travel time of the next link varies. This leads to the above formulation. As shown above, the number of decision variables in the linear program dramatically increases with respect to the cardinality of the dependent link set ψ and the number of preceding nodes set to each link $(j,k) \in \psi$.

In general, the actual number of distinct links $X_{i,(j,k)}$ in a dependent stochastic network is found as follows,

$$|\widetilde{\psi}| + \sum_{(j,k)\in\psi} |A(j,k)| \tag{3.25}$$

where the notation $|S|$ is the cardinality of set S. Again, the number of decision variables $X_{i,(j,k)}$ increases with respect to the product of $|A(j,k)| \times |\psi|$.

For example, we consider the following undirected graph in Figure 3.5. This graph consists of 5 nodes and 8 arcs as shown.

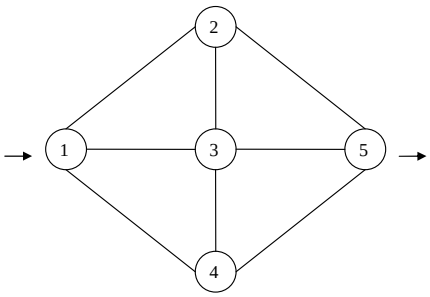

Figure 3.5 A dependent stochastic network

For this graph, the dependent link set ψ and independent links set $\widetilde{\psi}$ are given as follows,

$$\psi = \{(2,5),(3,5),(4,5),,(2,3),(3,2),(3,4),(4,3)\}$$
$$\widetilde{\psi} = \{(1,2),(1,3),(1,4)\}$$

The preceding nodes set for link $(3,5)$, for example, is $A(3,5) = \{2,1,4\}$. The number of decision variables for this problem is,

$$\begin{aligned} |\widetilde{\psi}| + \sum_{(j,k)\in\psi} |A(j,k)| &= 3 + 14 \\ &= 17 \end{aligned}$$

This LP approach requires us to enumerate all possible distinct links and compute all the dependent mean random travel times $c_{i,(j,k)}$ of dependent link (j,k) in advance. While this approach gives us the exact solution, this becomes very intractable as the number of links and nodes increases. Therefore, in the next sub-

section, a heuristic algorithm is developed to find the expected shortest path of a dependent network.

3.4.3 The K-shortest path heuristic

The basic idea of this heuristic is to identify the first dominating path set in terms of independent expected travel time and then, determine the dependent expected shortest path among them depending on the dependency approach.

In order to find the dominated path set, we use the K-shortest path algorithm based on independent expected travel time. Before proceeding to this algorithm, we need to understand this K-shortest path algorithm. Next, we briefly describe the algorithm.

K shortest path algorithm

We present the double sweep algorithm developed by Shier [3] in 1979. This algorithm finds the K shortest path lengths between a specified node and all other nodes in a graph. This algorithm consists of exclusively generalized addition and minimization set operations. These operations are performed on sets of K distinct numbers representing the lengths of paths or arcs.

Initially, for a graph $G(N, E)$ where $|N| = n$, we construct an initial estimate vector of K shortest distance from the source node to all nodes. The double sweep algorithm method successively reduces these estimates until an optimal set of distances is found using two set operations; generalized minimization and addition.

First, let R^K denote the set of all vectors $(d_1, d_2, ..., d_K)$ with the property that $d_1 < d_2 < .... < d_K$. Thus,the components (estimates of distance) of a member of R^K are distinct and arranged in ascending order.

Let $a = (a_1, a_2, ..., a_K)$ and $b = (b_1, b_2, ..., b_K)$ be two members of R^K. Then, generalized minimization, denoted by $\oplus$, is defined as

$$a \oplus b = min_k\{a_i, b_i : i = 1, 2, ..., K\} \tag{3.26}$$

where $min_K(X)$ means the K smallest distinct members of the set X.

Generalized addition, denoted by $\otimes$ is defined as

$$a \otimes b = min_K\{a_i + b_j : i, j = 1, 2, ..., K\} \tag{3.27}$$

The following example aids in understanding generalized minimization and addition. Suppose there are two vectors, a = [-4,0,1,∞] and $b = [1, 7, 8, 9]$. Clearly the two vectors are arranged in ascending order. In this case the generalized minimization is,

$$a \oplus b = [-4, 0, 1, 7]$$

The generalized addition can be easily found by using the cross sum of both of elements in vector a and b,

b$\otimes$ a	-4	0	1	∞
1	-3	1	2	∞
7	3	7	8	∞
8	4	8	9	∞
9	5	9	10	∞

The set of cross sums is therefore,

$$[-3, 1, 2, \infty, 3, 7, 8, \infty, 4, 8, 9, \infty, 5, 9, 10, \infty] \text{ or}$$
$$[-3, 1, 2, 3, 4, 5, 7, 8, 9, 10, \infty] \text{ so}$$
$$b \otimes a = [-3, 1, 2, 3]$$

Thus, the generalized minimization defines a set formed with the elements of two given vectors of equal dimension used to construct a third vector of the same dimension by putting elements in the set in strictly increasing order. The generalized addition defines a set formed with the cross sum of the elements of two given vectors

of equal dimension, and then constructs a third vector of the same dimension, with the minimal element of the set of cross sums in the first position.

Given three vectors such that,

$$a = [a_1, a_2, ..., a_K] \ , \ f = [0, \infty, \infty, ..., \infty] \ , \ \text{v}=[\infty, \infty, \infty, ..., \infty].$$

Clearly the following relations hold:

$$\text{a} \oplus \text{v} = \text{a}$$
$$\text{a} \otimes \text{v}= \text{v}$$
$$\text{a} \otimes \text{f} = \text{a}$$

Now, let $d_{ij}^0 = (d_{ij1}^0, d_{ij2}^0, ..., d_{ijK}^0) \in R^K$ denote the initial lengths of the K shortest paths from node i to node j. If two paths from node i to node j have the same length, then this length appears only once in d_{ij}^0. If there are less than K paths from i to j, then fill up the remaining components with ∞. Let $d_{ij}^l = (d_{ij1}^l, d_{ij2}^l, ..., d_{ijK}^l)$ $\in R^K$ denote the K shortest distinct path lengths from node i to j acquired at lth iteration. Let D^l denote the matrix whose i,jth element is the d_{ij}^lvector. Lastly, let $d_{ij}^* = (d_{ij1}^*, d_{ij2}^*, ..., d_{ijK}^*) \in R^K$ denote the K shortest path lengths from node i to j. These path lengths are called the *optimal path lengths.*

Consequently, let D^* denote the matrix whose i,jth element is the d_{ij}^* vector. This double sweep algorithm starts with an initial solution of K shortest path lengths from node i to j such as $d_{ij}^0 = (0, \infty, \infty, ..., \infty)$.

At each iteration, it employs a sequence of alternating forward and backward iterations. During the forward iteration, the nodes are examined in the order $1, 2, ..., n$, while the nodes are examined in the order $n, ..., 2, 1$ during the backward iteration. Moreover, when examining a node j during the forward iteration, only arcs (i, j) with $i < j$ are processed. Similarly, only arcs (i, j) with $i > j$ are processed during the backward iteration. By processing, we mean that the current K-vector for node j will be improved if possible by a path to node j which extends

first node i and then uses the arc (i, j). More precisely, if any of the quantities $\{d_{1im} + c_{ij} : m = 1, .., K\}$ where m is the order of shortest paths vector, provides a smaller path length than any one of the tentative K shortest path lengths in d^l_{1j} at the lth iteration, the current K vector d^l_{1j} is updated by inclusion of this smaller path length [3].

To attain the above forward and backward sweep improvement, we need to construct two matrices L and U. Let matrix L be formed from matrix D^0 by replacing every component of every element d^0_{ij} by ∞ whenever $i \leq j$. Let matrix U be formed from matrix D^0 by replacing every component of every element d^0_{ij} by ∞ whenever $i \geq j$. So matrices L and U are the lower and upper triangular portions of D^0.

The double sweep algorithm is initiated with any estimate $d^0_{ij} = (d^0_{ij1}, d^0_{ij2}, .. , d^0_{ijK})$ of $d^*_{ij} = (d^*_{ij1}, d^*_{ij2}, .., d^*_{ijK})$ in which each value is not less than the corresponding optimal value. For example, all values could be set equal to ∞ except $d^0_{111} = 0$. The algorithm computes new improved (reduced) estimates of d^*_{ij} by processing d_{ij} if any of the K values in $d^0_{1j} \otimes d^0_{ji}$ are less than any of the K values in the current estimate d^0_{1i}. If so, the smallest K values are chosen. This process implemented for all j incident to node i, yielding a new improved estimate d^0_{1i} of $d1i^*$ (optimal).

The termination condition is that two successive estimates $(d^i_{11}, d^i_{12}, ..., d^i_{1n})$ and $(d^{i+!}_{11}, d^{i+1}_{12}, ..., d^{i+1}_{1n})$ are the same in every component for $i \geq 1$. Then this final estimate is the optimal value. The detail of this algorithm can be found in Shier [3]. Following is a brief description of the double sweep algorithm:

Step 0 Initialization. Let the initial estimate $d^0_1 = (d^0_{11}, d^0_{12}, ..., d^0_{1n})$ of d^*_1 consist of values that equal or exceed the corresponding optimal values. For all diagonal element vectors, $d^0_{ii} = [0, \infty, \infty, ..., \infty]$.

Step 1 Given an estimate d^{2r}_1 of d^*_1, calculate new estimates d^{2r+1}_1 and d^{2r+2}_1 as follows:

Backward sweep:

$d_1^{2r+1} = d_1^{2r+1} \otimes \oplus L \oplus d_1^{2r}$

Forward sweep:

$d_1^{2r+2} = d_1^{2r+2} \otimes \oplus U \oplus d_1^{2r+1}$ $(r = 0, 1, 2, ...)$.

Step 2 If two successive estimates d_1^{t-1} and d_1^t are identical for $t > 1$, terminate, otherwise go to step 1.

Note that the multiplication operation in the first part in both sweep consists of generalized minimization of each set, $d_{ij} \oplus L_{ji}$ and generalized addition of all such sets. Once we found the K shortest path lengths from the source node to all nodes in matrix D^*, we need to find each path corresponding to the length of the path. This tracing procedure is nontrivial since we cannot trace the path at each iteration because of arbitrarily larger than optimal estimate values.

Thus, tracing can be done once after we have found the optimal K shortest path lengths. Suppose we want to find the pth shortest path from the source node to some node j. The tracing procedure follows:

Step 1 Identify the incident node (penultimate node) to j where any of K shortest path lengths in the distance vector satisfies Equation (3.28),

$$d_{1jp}^* = d_{1it}^* + d_{ij1}^0 \tag{3.28}$$

where d_{1it}^* is the t^{th} shortest path length, $t \leq p$ corresponding to node j. d_{ij1}^0= the arc length from node i to j.

Step 2 Repeat Step 1 until we reach the source node.

However, in practice we can encounter a tied situation choosing the penultimate node i or may have a cycle path. These cases need to be resolved in the implementation. One of the cycle reduction techniques is presented in Appendix B. Finally using

this K shortest path algorithm, we present the dependent expected shortest path heuristic next.

Dependent expected shortest path heuristic

Based on the K shortest paths algorithm, we develop the following K-shortest path heuristic to find the dependent expected shortest path for a stochastic network:

Step 0 Construct the matrix $A = [a_{ij}]$ where a_{ij}=$E[T(x_{ij})]$, the mean travel time on arc (i, j) using Equations (2.10) and (2.14).

Step 1 Find the K shortest paths based on the above matrix A.

- Finding K shortest paths from source node to sink node using double sweep and tracing procedures.
- Restore the K sets of paths from source to sink node in a list P.

Step 2 Set $i = 1$; start with the longest path from list P and set it to p^*. Calculate the dependent expected travel time t^* based on one of two approaches: expected terminal distribution or expected terminal state.

Step 3 If $i > K$, p^* is the optimal path with t^*. Otherwise goto Step 4.

Step 4 Set $i = i + 1$, take the i^{th} path from P and set it to p^i. If the calculated dependent expected travel time value $t^i < t^*$, then $p^* = p^i$. Go to Step 3.

At Step 1, we use Shier's K-shortest path algorithm [12]; called *double sweep algorithm*, based on the independent expected travel time of each link. The efficiency of this algorithm relies on the K value, the number of paths to be selected. Obviously, as the value of K increases, the chance of finding the optimal dependent expected shortest path increases, but required computational effort also increases.

The underlying concept in this heuristic is to find a subset of paths $P_K \subset P$ based on independent mean travel time. This concept is similar to Fu and Rilett's work [20] to find the expected shortest path in dynamic networks. The authors

obtained good results even with small K in a problem with 1400 nodes. We demonstrate the effect of the value of K on finding the optimal dependent shortest path in Chapter 4.

3.5 Algorithms for the Stochastically Shortest Path

In this section, we analyze methods to compute the stationary travel time distribution on any given path in the stochastic network and develop the algorithm to find the stochastically shortest path. The first subsection considers the case of independent link travel times.

3.5.1 Method of Convolution

In order to find the total travel time distribution of any given path, we have to sum all the travel time random variables on each link of the path. This problem reduces to finding the distribution of the sum of random variables.

Let $T(x_{ij})$ denote the random travel time to traverse link (i, j). Let φ denote the set of all paths from the source node to sink node k. Then, $\varphi = \{p : p \text{ is a path from node 1 to node } k\}$. Further, let γ_p be the total random travel time of path $p \in \varphi$. That is $\gamma_p = \sum_{i,j \in p} T(x_{ij})$. Finally our objective is to find the cumulative distribution function of γ_p,

$$F_{\gamma_p}(t) = P\{\gamma_p \leq t\}.$$

There are numerous methods to find this seemingly simple sum of random variables. Since we first assume independent link travel time, convolution is one method to solve the problem.

For a given path $p \in \varphi$, Figure 3.6 depicts the travel time random variable of each link. Let us seek the survival function of the total travel time of the above path, namely $G_{\gamma p}(t) = 1 - F_{\gamma p}(t)$. For path $p \in \varphi$, the distribution of γ_p is the convolution

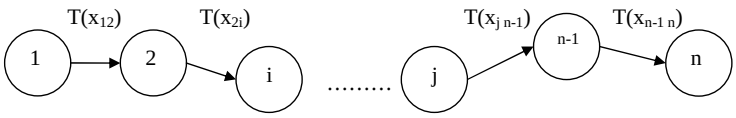

Figure 3.6 Convolution of all link travel times along a path.

of all travel time distributions for each link $(i, j) \in p$. That is,

$$G_{\gamma_p}(t) = G_{T(x_{12})}(t) * G_{T(x_{2i})}(t) * \cdots * G_{T(x_{n-1n})}(t) \tag{3.29}$$

where the survival function of the random travel time of link (i, j) is defined as,

$$G_{T(x_{ij})}(t) = \sum_{h \in S_{ij}} H_h(x_{ij}, t)$$

Taking the Laplace transformation of Equation (3.29) with respect to t with complex s_2,

$$\begin{aligned} G^*_{\gamma_p}(s_2) &= G^*_{T(x_{12})}(s_2) \cdot G^*_{T(x_{2i})}(s_2) \cdot \cdots \cdot G^*_{T(x_{n-1n})}(s_2) \\ &= [H(x_{12}, s_2)e] \cdot [H(x_{2i}, s_2)e] \cdot \cdots \cdot [H(x_{n-1n}, s_2)e] \end{aligned} \tag{3.30}$$

where $H(x_{ij}, s_2)$ is the vector of $H_h(x_{ij}, s_2)$, $h \in S_i j$. If we recall Theorem 2.1 in Chapter 2, in order to find the solution of $[H(x_{ij}, t)]$ we took the transformation twice with respect to t and x_{ij}, respectively (Kharoufeh and Gautam [25]). Hence, using the solution of $H(x_{ij}, t)$ found by twice transformation, Equation (3.30) can be computed as follows,

$$\begin{aligned} G^*_{\gamma_p}(s_2) &= [H(x_{12}, s_2)] \cdot [H(x_{2i}, s_2)] \cdots [H(x_{n-1n}, s_2)] \\ &= \mathcal{L}^{-1}|x_{12}[s_1^{-1}\tilde{H}^*(s_1, s_2)e] \cdot \mathcal{L}^{-1}|x_{2i}[s_1^{-1}\tilde{H}^*(s_1, s_2)e] \cdots \\ &\quad \cdot \mathcal{L}^{-1}|x_{n-1n}[s_1^{-1}\tilde{H}^*(s_1, s_2)e] \end{aligned} \tag{3.31}$$

where $\mathcal{L}^{-1}|_{x_{ij}}$ is the Laplace transform inversion with respect to the length of link (i,j). This apporach becomes intractable even with a small number of convolutions.

Suppose a path p where the link set of path p is given as $E(p) = \{(1,2),(2,3)\}$. We seek the convolution of two independent random travel times of links $(1,2)$ and $(2,3)$. This is computed as follows,

$$\begin{aligned} G^*_{\gamma_p}(s_2) &= G^*_{T(x_{12})}(s_2) \cdot G^*_{T(x_{23})}(s_2) \\ &= [H(x_{12},s_2)] \cdot [H(x_{23},s_2)] \\ &= \mathcal{L}^{-1}|_{x_{12}}[s_1^{-1}\tilde{H}^*(s_1,s_2)e] \cdot \mathcal{L}^{-1}|_{x_{23}}[s_1^{-1}\tilde{H}^*(s_1,s_2)e] \\ &= \begin{array}{l} \mathcal{L}^{-1}|_{x_{12}}[s_1^{-1}\tilde{B}(s_1)(s_1V_{12}+s_2I-Q_{12})^{-1}e]\cdot \\ \mathcal{L}^{-1}|_{x_{23}}[s_1^{-1}\tilde{B}(s_1)(s_2V_{23}+s_2I-Q_{23})^{-1}e] \end{array} \end{aligned} \tag{3.32}$$

where V_{ij}, Q_{ij} are the diagonal velocity matrix and Q matrix of link (i,j), respectively. Once we compute Equation (3.32), we still have to inverse Laplace transform one more time to compute the distribution of $G_{\gamma_p}(t)$ as follows,

$$G_{\gamma_p}(t) = \mathcal{L}^{-1}\{G^*_{\gamma_p}(s_2)\}$$

In fact, this approach requires $2n$ Laplace transformations and n times one dimensional Laplace inversion as the number of links in a path, n increases. Thus, total of $3n$ Laplace transformation and inversions are needed. This computation is very intractable and not feasible in reality where certain paths often have hundreds of links.

We can attain computational feasibility using a research result [25] that the travel time distribution of this CTMC model can be parameterized with acceptable statistical criteria. We present this result in the next section.

3.5.2 Parametric Approximations

In this section we review some results from Kharoufeh and Gautam [24] that parameterized the distribution of travel time using the environment process model. They assumed the normal distribution as an approximate parametric distribution of random travel time. As shown in Chapter 2, there are two types of parameters of normal approximated distributions; transient parameters and asymptotic parameters.

At this point, we need to recall the definition of the transiently parameterized normal distribution and the asymptotically parameterized normal distribution. The transient normal distribution of random travel time is defined as a parameterized normal distribution, $N(\mu, \sigma^2)$ with transient parameters where

$$\mu = m_1(x) = \mathcal{L}^{-1}[s_1^{-1}\tilde{B}(s_1)(s_1 V - Q)^{-1}e]$$
$$\sigma^2 = m_2(x) - m_1(x)^2$$

The asymptotic normal distribution of random travel time is defined as a parameterized normal distribution, $N(\mu, \sigma^2)$with asymptotic parameters where

$$\mu = m_1(x) = \frac{x}{pv}$$
$$\sigma^2 = -\frac{2x}{pv}\sum_{i=2}^{K}\frac{1}{\eta_i}\frac{(pr_i)(l_i V^{-1}e)}{l_i r_i}$$

These are defined in Chapter 2 (Equations (2.14) and (2.16)). Thus, the transient normal distribution is based on the transient behavior of a CTMC of environment process. The asymptotic distribution is based on the asymptotic behavior of the CTMC.

Once we have parameterized the travel time distribution of each individual link as a normal distribution, we can compute the path distribution more simply than

the convolution approach as follows:

$$\gamma_p \sim N(\sum_{(i,j)\in p} \mu_{ij}, \sum_{(i,j)\in p} \sigma_{ij}^2) \tag{3.33}$$

where μ_{ij} and σ_{ij}^2 are the mean and the variance of link $(i,j) \in p$, respectively. This result is based on the fact that the sum of normal random variables is again normal with mean equal to the sum of the means, and variance equal to the sum of the variances.

The simple algorithm proceeds as follows:

Step 0 Initialization.

- Distance matrix. $X = [x_{ij}]$ where x_{ij} is the length of link (i,j) if nodes i,j are adjacent and 0 otherwise.
- CTMC sample space S_{ij} of each link (i,j).
- Generator matrix Q_{ij} for each link (i,j).
- Velocity matrix V_{ij} for each link (i,j).

Step 1 Compute the parameters of the normal path distribution.

- Find the asymptotic $m_1(x_{ij})$ and transient $m_1(x_{ij})$ using Equations (2.14) and (2.10) respectively.
- Find the asymptotic $\sigma(x_{ij})$ and transient $\sigma(x_{ij})$ using Equations (2.16) and (2.12), respectively.
- Let $m_1(x_p) = \sum_{(i,j)\in p} m_1(x_{ij})$, $\sigma(x_p) = \sum_{(i,j)\in p} \sigma(x_{ij})$ where x_p is the total length of path $p \in \varphi$.

Step 2 Then, path p travel time $\gamma_p \sim N\left(m_1(x_p), \sigma(x_p)\right)$.

Because of the normality assumption, we can approximate the total random travel time distribution of any path in an independent stochastic network.

In the next subsection, we utilize this parametric approximation result to find the stochastically shortest path.

3.5.3 Finding the Stochastically Shortest Path

The above result of the normality of the travel time of a path enables us to utilize some path optimality measures. First, suppose we are interested in finding the optimal path p^* such that,

$$p^* = argmax_{p \in \varphi} P\{\gamma_p \leq \gamma_q\} \tag{3.34}$$

where $p, q \in \varphi$. In other words, Equation (3.34) is stated as finding the stochastically shortest path over other paths. Thanks to the normality of our model, the above problem turned out to be equivalent to finding the independent expected shortest path as shown in proposition 3.4.

Proposition 3.4 *Assuming all link travel times are independent and normally distributed, the stochastically shortest path problem is equivalent to the independent expected shortest path problem, i.e,*

$$p^* = argmax_{p \in \varphi} P\{\gamma_p \leq \gamma_q\} \Leftrightarrow p^* = min_{p \in \varphi} E[\gamma_p] \tag{3.35}$$

Proof. Suppose at stage $n-1$ in dynamic programming, to find a link to be included in the optimal path p^* and there is a candidate links set h where $h = \{i :$ i is the incident link to the n-1 stage link$\}$. Let p_{n-1} be the link set chosen at stage $n-1$ where $p_{n-1} = \{j : j \in p$ at stage n-1$\}$ and $\gamma_{p^{n-1}}$ be the random travel of path p up to stage $n-1$.

Our objective is to find the link i such that

$$p^* = argmax_{i \in h} P\{\gamma_{p^{n-1},i} \leq \gamma_{p^{n-1},j} : \forall j \in h\}$$

where $\gamma_{p^{n-1},i}$ is random travel time of path p with link i at stage n.

Then, the following equations hold.

$$\begin{aligned} P\{\gamma_{p^{n-1},i} \le \gamma_{p^{n-1},j}\} &= P\{\gamma_{p_{n-1}} + T(x_{n-1,i}) \le \gamma_{p_{n-1}} + T(x_{n-1,j})\} \\ &= P\{\gamma_{p_{n-1}} - \gamma_{p_{n-1}} + T(x_{n-1,i}) - T(x_{n-1,j}) \le 0\} \\ &= P\{T(x_{n-1,i}) - T(x_{n-1,j}) \le 0\} \\ &= P\{Z_{i,j} \le 0\} \end{aligned} \tag{3.36}$$

where $Z_{i,j} = T(x_{n-1,i}) - T(x_{n-1,j})$. We have to find the link i such that $max P\{Z_{i,j} \le 0 \forall j \in h\}$. Since $Z_{i,j} \sim N\left(m_1(x_{n-1,i}) - m_1(x_{n-1,j}), \sigma^2_{T(x_{n-1,i})} + \sigma^2_{T(x_{n-1,j})}\right)$, if we compare all possible pairs of links in candidate link set h, clearly the following relationship holds,

Proposition 3.5 *For two normal random variables $\gamma_{p^{n-1},i}$ and $\gamma_{p^{n-1},j}$,*

$$P\{\gamma_{p^{n-1},i} \le \gamma_{p^{n-1},j}\} \ge P\{\gamma_{p^{n-1},j} \le \gamma_{p^{n-1},i}\} \quad \textit{iff} \quad m_1(x_{n-1,i}) \le m_1(x_{n-1,j}).$$

Proof.

$$\begin{aligned} P\{\gamma_{p^{n-1},i} \le \gamma_{p^{n-1},j}\} &\ge P\{\gamma_{p^{n-1},j} \le \gamma_{p^{n-1},i}\} \\ P\{Z_{i,j} \le 0\} &\ge P\{Z_{j,i} \le 0\} \end{aligned} \tag{3.37}$$

The distribution of $Z_{i,j}$ and $Z_{j,i}$ are respectively,

$$\begin{aligned} Z_{i,j} &\sim N\left(m_1(x_{n-1,i}) - m_1(x_{n-1,j}), \sigma^2_{T(x_{n-1,i})} + \sigma^2_{T(x_{n-1,j})}\right) \\ Z_{j,i} &\sim N\left(m_1(x_{n-1,j}) - m_1(T(x_{n-1,i})), \sigma^2_{T(x_{n-1,j})} + \sigma^2_{T(x_{n-1,i})}\right) \end{aligned}$$

As shown the variances of two distribution are the same. Hence, continuing Inequality (3.37),

$$\begin{aligned} P\{Z_{i,j} \leq 0\} \geq P\{Z_{j,i} \leq 0\} \quad &\Leftrightarrow \quad m_1(x_{n-1,i}) - m_1(x_{n-1,j}) \leq m_1(x_{n-1,j}) - m_1(x_{n-1,i}) \\ &\Leftrightarrow \quad m_1(x_{n-1,i}) \leq m_1(x_{n-1,j}). \end{aligned}$$

Finally the following dual relationship is drawn,

$$max_i P\{Z_{i,j} \leq 0 : \forall j \in h\} \Leftrightarrow min_i\{m_1(x_{n-1,i}) : i \in h\} \qquad \text{Q.E.D.}$$

Hence, due to the normality property of our model, finding the optimal shortest path defined in Equation (3.34) is equivalent to finding the independent expected shortest path. Further, this result is one of the major developments we found in this research using the environment process to model a network where the normality property of the environment process leads to the above result.

In this chapter, we have developed various methodologies to find the expected shortest path in a stochastic and dynamic environment. These methodologies range from a basic independent stochastic network to successively dependent stochastic networks. We also develop the path total travel time distribution through parametric approximations [25] with an independence assumption. Further, we developed a stochastically shortest path optimality measure and found that the approximated normal distribution of each link makes the stochastically shortest path equivalent to the expected shortest path in an independent stochastic network environment. In Chapter 4, we experiment and analyze these methodologies with numerical examples.

4. Numerical Experimentation and Results

The primary objective of this chapter is to demonstrate the methodologies of Chapter 3. If the initial distribution does not affect the mean travel time of the link significantly, then our dependence methodologies through the initial distribution may note justify the additional computational effort as compared to the independence methodology. Consequently, one should consider when the dependence is significant. For the purpose of validation of the K shortest path heuristic, we also study the robustness of the K shortest path heuristic for dependent stochastic networks with respect to variation of the K parameter along with CTMC behavioral aspects: transient approach and asymptotic approach.

First, we determine when the dependence on the initial distribution of the CTMC makes a difference through analyzing the impact of the initial distribution on the expected travel time of a link with some notional Q matrices. Then, we present a sample problem imposing the dependence. After we address the dependence analysis, we present two formal numerical examples and apply the independence and dependence methodologies, respectively, to the problem as well as the total path travel time distribution. In the first formal numerical example, Q matrices have erratic structure and we solve the problem as well as study the complexity of the K-shortest path heuristic. In the second numerical example, we present an example problem where the sample space of the CTMC of all links are distinct, and the Q matrices are more moderate.

4.1 Dependence Analysis

In this section, we analyze the impact of varying the initial distribution on mean travel time of an individual link. Then, we experiment with the dependence methodologies with a simple small problem before extending to a general network problem case.

4.1.1 Dependence and the transient period

In our dependent network models, the dependence of successive links lies on the initial distribution of the CTMC governing the vehicle velocity. In other words, our two dependence methods rely on the preceding CTMC's termination conditions; either expected terminal distribution (dependent I) or expected terminal state (dependent II) in Section 3.3.

This leads to the following natural question: How much does the variation of the initial distribution of the CTMC governing link x_{ij} impact the final expected travel time of the subsequent link? Intuitively, this question directly leads to how fast the CTMC converges to steady-state. The faster the CTMC converges to steady-state, the faster the impact of the initial state of CTMC vanishes [15]. Once the impact of initial state distribution vanishes, our dependence concept would be thought to not make much difference. Thus, naturally this issue leads to analysis of validity of our dependence concept through the initial distribution impact on mean travel time.

If the transient period of the CTMC is shorter than the expected travel time, then obviously the initial state distribution is expected to not impact the expected travel time because the CTMC already converged to steady-state before the expected travel time. Consequently, our dependence concept considers only the case where this transient period is thought to be long enough to make the initial state distribution have an impact. In order to analyze the transient period duration with respect to various initial distributions, we need to study the transient probability behavior.

The CTMC transition probability matrix $P(t)$ is the unique solution to

$$\frac{d}{dt}P(t) = P(t)Q = QP(t)$$

where $p_{ij}(t) = P\{Z(t) = j | Z(0) = i\}$ and $P(t) = [p_{ij}(t)]$. To solve above the differential equation, numerous methods are available [15] from N^{th} order differential

equation solution method to the uniformization method. However, since we seek the matrix solution and considering the possibility of numerical instability for numerical inversion of Laplace transforms, we instead use matrix exponentiation as follows [15].

$$\begin{aligned} P(t) &= \exp\{Qt\} \\ &= A\exp\{Dt\}A^{-1} \end{aligned} \tag{4.1}$$

where $D = \text{diag}(\mu_1, \mu_2, ..., \mu_n)$ whose μ_i denotes the ith eigenvalue of Q and A is the matrix of right eigenvectors. Then, if we define the marginal state probability as follows,

$$\alpha(t) = [\alpha_j(t)]$$

where $\alpha_j(t) = P\{Z(t) = j\}$, the marginal state probability vector is found as,

$$\alpha(t) = z_0 P(t). \tag{4.2}$$

where z_0 is the initial distribution of the CTMC.

Consider the following example of a two state CTMC with $S = \{1, 2\}$ [15],

$$Q = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}.$$

The conditional probability matrix $P(t)$ is given as,

$$P(t) = \begin{bmatrix} \frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} + \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu}(1 - e^{-(\lambda+\mu)t}) \\ \frac{\mu}{\lambda+\mu}(1 - e^{-(\lambda+\mu)t}) & \frac{\lambda}{\lambda+\mu} + \frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t} \end{bmatrix}.$$

The marginal state 1 probability, $P\{Z(t) = 1\}$, denoted by $\alpha_1(t)$ is derived via Equations (4.1) and (4.2) as follows,

$$\alpha_1(t) = z_{0_1}\left(\frac{\lambda}{\lambda+\mu}e^{-}(\lambda+\mu)t + \frac{\mu}{\lambda+\mu}\right) + z_{0_2}\left(\frac{\mu}{\lambda+\mu}(1 - e^{-}(\lambda+\mu)t)\right). \tag{4.3}$$

where z_{0_i} denote the initial probabilities, $P\{Z(0) = i\}$, $i \in S$. The impact of the initial distribution z_{0_1}, z_{0_2} clearly decreases asymptotically and the steady-state distribution becomes independent of the initial distribution as follows,

$$\lim_{t\to\infty} \alpha_1(t) = \lim_{t\to\infty} p_{12}(t) = \lim_{t\to\infty} p_{21}(t). \tag{4.4}$$

Equation (4.4) indicates that the transition probability and marginal probability asymptotically converge to the same value, removing the dependence on the initial distribution z_0.

Consequently, the expected mean travel time of individual link becomes independent of the initial distribution as t approaches infinity. This was shown by Kharoufeh and Gautam [24] as follows.

$$\lim_{s_1\to 0} s_1[z_0(s_1V - Q)^{-1}e] = (pv)^{-1} \tag{4.5}$$

Consequently, as $t \to \infty$ and $x \to \infty$ [24],

$$\frac{m_1(x)}{x} \longrightarrow \frac{1}{pv}. \tag{4.6}$$

As t approaches infinity, since the $[P(t)] \to [p]$, the transient mean travel time becomes asymptotic mean travel time which is independent of the initial distribution. Therefore, our dependence methodologies rely on the transient period of CTMC of the individual link where the initial distribution will produce a different mean travel time value.

Next, we consider the initial distribution impact on the duration of the transient period through a notional numerical example.

4.1.2 Initial distribution impact

The CTMC transient period study mainly involves the Q matrix and the initial distribution. Instead of a formal theoretical study of the factors affecting the duration of transient period of the CTMC, in this effort we only study the analysis of mean travel time as it depends on the initial distribution and Q matrix to model the dependent stochastic network. Particularly, the duration of the transient period is directly related to the Q matrix structure, specifically each transition rate in the Q matrix. Thus, this subsection provides an empirical sensitivity analysis of the mean travel time to the initial distribution of an individual link with a given Q matrix structure. We need to note again the objective of this sensitivity experiment is to identify when imposing the dependence would make a significant difference in mean travel time with regard to the structure of the Q matrix.

Consider the following small problem. We generated three different representa-

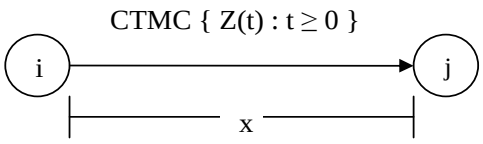

Figure 4.1 Sample network link (i, j).

tive Q matrices $(Q(1), Q(2), Q(3))$ and various initial distributions (Table 4.1). The objective of this experiment is to identify the Q matrix structure whose transient period is the longest, or equivalently whose impact of initial distribution is the longest so that our dependence methodologies would be valid enough to make a critical difference in mean travel time. The experiment configuration is given in Table 4.1. In this experiment we only focus on the behavior of CTMC and do not yet consider the distance of the above link, x. For the Q matrix, we made two distinct structures, an erratic and an uniform structure. For $Q(1)$, we purposely make state 1 a semi-absorbing state where once the CTMC enters this state, it rarely exits. Compared to the other state's transition rate range, $100 \sim 200$, state 1's transition rate falls

Table 4.1 Sensitivity experiment configuration.

Factors	Q matrix variation	Q(1)(Erratic structure)	Q(2)(Uniform structure)	Q(3)(Erratic structure)
		$\begin{bmatrix} -0.8538 & 0.0850 & 0.7688 \\ 159.3771 & -291.0460 & 131.6689 \\ 123.3395 & 100.8422 & -224.1818 \end{bmatrix}$	$\begin{bmatrix} -349.6759 & 171.4797 & 178.1962 \\ 123.7565 & -250.0779 & 126.3214 \\ 171.3785 & 197.7600 & -369.1385 \end{bmatrix}$	$\begin{bmatrix} -365.0155 & 184.8057 & 180.2098 \\ 166.8306 & -348.8950 & 182.0644 \\ 0.6416 & 0.3063 & -0.9479 \end{bmatrix}$
	Initial distribution variation	$z_0(1)$	$z_0(2)$	$z_0(3)$
		[1 , 0 , 0]	[0.33 , 0.33 , 0.34]	[0 , 0 , 1]
Response	Marginal probability & Transient Period	• The marginal probability of state 1 : P{ Z(t) = 1 } • Transient period : | P { Z(t) =1 } - P_1 | < 0.01		

within $0 \sim 1$ which makes the $Q(1)$ structure erratic. State 3 of $Q(3)$ matrix is also made in a similar manner. For $Q(2)$, all transition rates are generated uniformly within $100 \sim 200$ which is referred to as uniform structure. This Q matrix structure plays an important role in terms of the initial distribution impact on mean travel time. The result is summarized in Figure 4.2 and Table 4.2.

Table 4.2 Transient period time under various governing CTMC.

Distinct Q matrix	Q(1)	Q(2)	Q(3)
Estimated Transient Period (min)	0.048	0.009	0.025

Each figure in Figure 4.2 represents the transient behavior of the marginal probability of state 1, $P\{Z(t) = 1\}$ with respect to time t depending on varying the initial distribution, $z_0(1)$, $z_0(2)$, $z_0(3)$. Clearly, the uniformly generated Q matrix ($Q(2)$) has the shortest transient period compared to the others. Thus, we can conclude that the initial distribution variation has the least impact period on CTMC with the $Q(2)$ matrix.

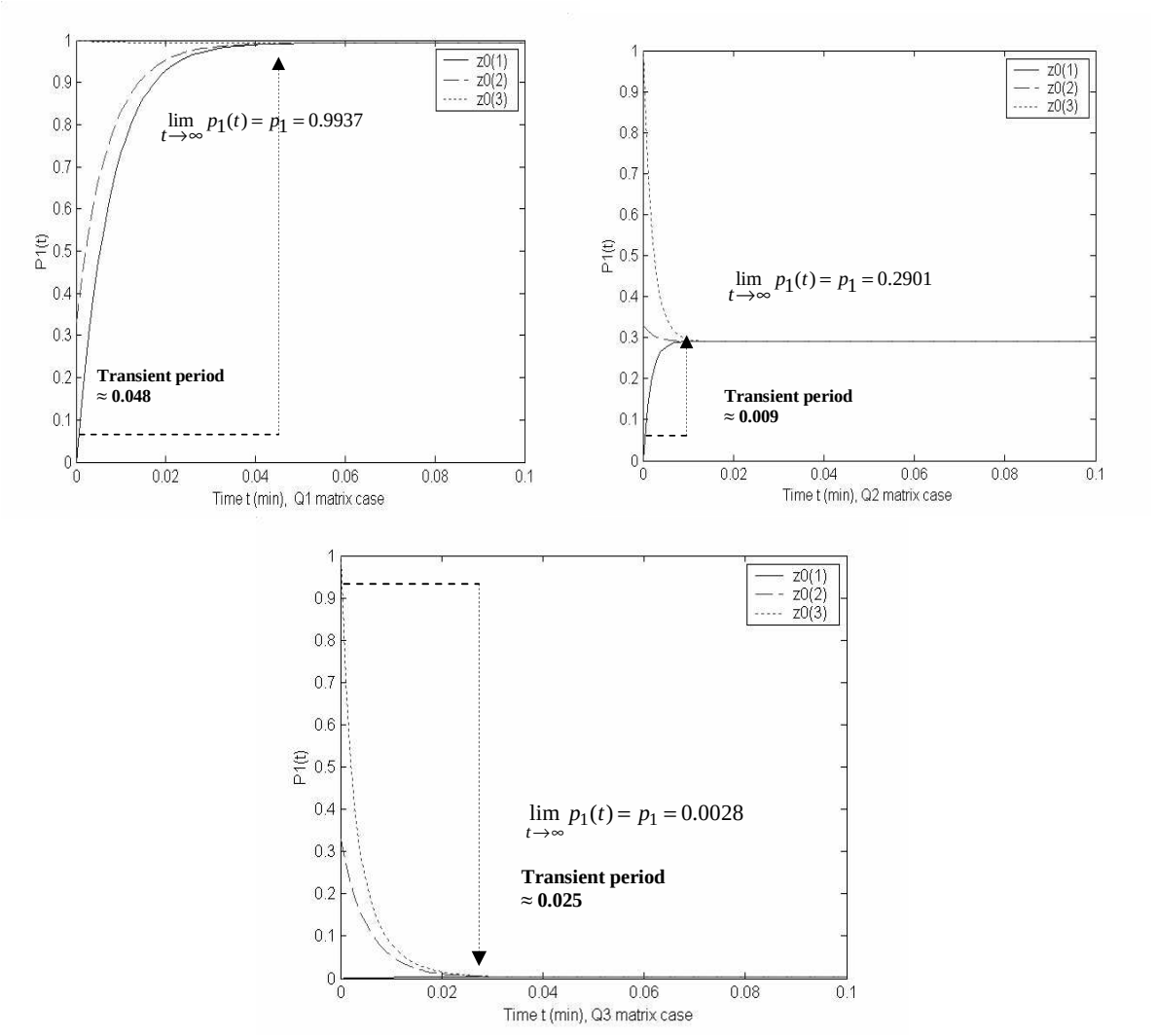

Figure 4.2 Graphical depictions of the transient period.

We conclude that the state of CTMC with Q matrix having uniform structure converges to steady state the fastest compared to other structures. Hence, if a link in a network has such an uniformized Q matrix, the dependency impact would be very negligible.

Next, we can legitimately infer that if the mean travel time, $m_1(x)$ exceeds the transient period of the other CTMCs with erratic Q matrix ($Q(1)$,$Q(3)$), the dependency impact could also be negligible. We verify this aspect of our model with respect to the variation of the length of a link.

Let us consider the CTMC with $Q(3)$ matrix case where we seek the length of link beyond which the variation of initial distribution has no effect on the mean travel time, $m_1(x)$. We design the experiment as follows (Table 4.3). Figure 4.3 describes

Table 4.3 Mean travel time variation experiment configuration.

Initial distribution variation	$z_0(1)$	$z_0(2)$	$z_0(3)$
	[1, 0 , 0]	[0.33 , 0.33 , 0.34]	[0 , 0 , 1]

the mean travel time $m_1(x)$ variation according to distinct initial distributions: $z_0(1)$, $z_0(2)$, $z_0(3)$. The different mean travel times do not seem to converge to one value.

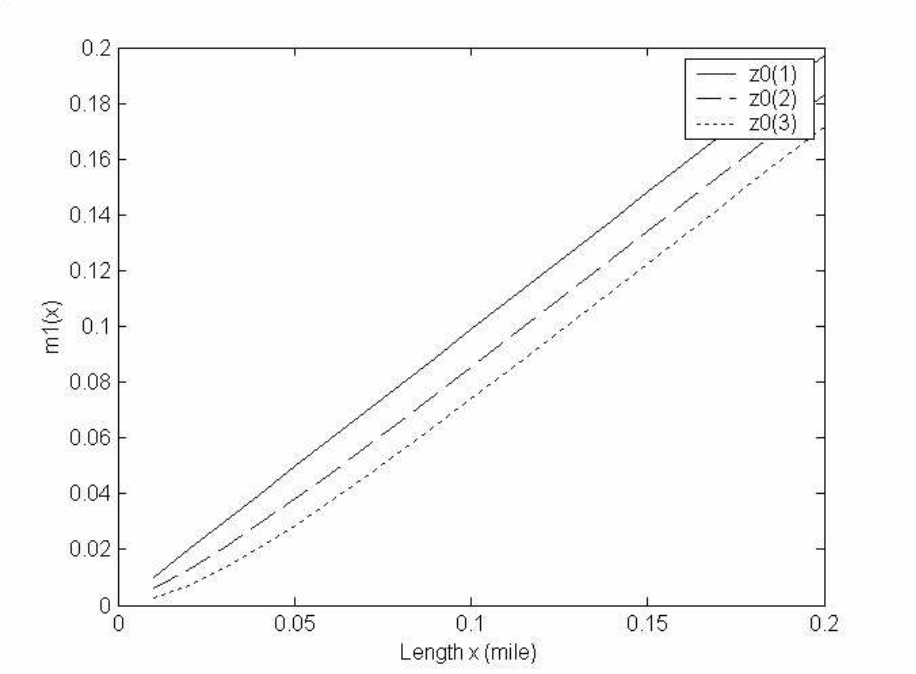

Figure 4.3 Mean travel time variation with respect to link length.

Rather, the difference between these different mean travel times seems to remain constant. In fact, numerically the differences very slowly increase as the distance of length increases. Further, this difference increases very negligibly within $\pm 6 \times 10^{-}6$ as the distance increases ten times which indicates the difference between three

distinct mean travel times and asymptotic mean travel time tends to be stationary with respect to the change of the distance.

Finally, in this experiment we come to conclude that the transient period mainly relies on the Q matrix structure. Specifically, the more erratic structure the Q matrix has, the longer the transient period tends to last.

Thus far, experiments indicate that when the Q matrix of a certain link in a path is erratic, the dependence on the preceding link makes a difference in mean travel time of that link. Further, we found that the difference in a mean travel time caused by dependence on preceding link tends to remain constant regardless of the length of the link.

These results give us insight as to when our dependence model would be more valid and makes a significant difference in finding the shortest path. In the next subsection, we experiment with a simple numerical example to see the difference between independence and dependence approaches.

4.1.3 Dependence experiment

In this subsection, we present the dependence experiment results with a small numerical problem. The objective of this subsection is to show how the dependence methods developed in Chapter 3 make difference in finding the shortest path with a small problem.

Let us consider the small network problem shown in Figure 4.4. The graph $G(N, E)$ whose node set is $N = \{1, 2, 3\}$ and arc set is $E = \{(1, 2), (1', 2'), (2, 3)\}$ has two distinct links between node 1 and 2; $(1, 2)$ and $(1', 2')$. Our objective is to find the shortest $1 \rightarrow 3$ path in this network.

The problem configuration is given in Table 4.4. We assume both sample space of CTMC of each link and the velocity function to be the same, $S = \{1, 2, 3\}$, $\nu = \{v_i : v_i = \frac{25}{i}, i \in S\}$, respectively. The generator matrices Q_{ij} of all links

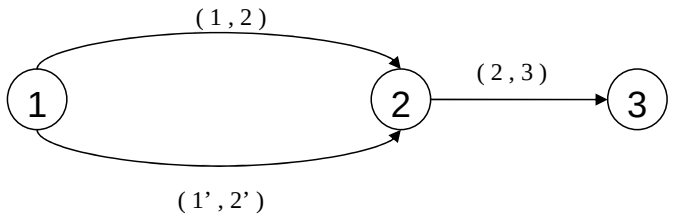

Figure 4.4 Dependence numerical example.

Table 4.4 Experiment Configuration.

Link / Factors	(1,2)	(1',2')	(2,3)
CTMC	$\{Z_{12}(T(x_{12}))\ ;\ T(x_{12}) \geq 0\}$	$\{Z_{1'2'}(T(x_{1'2'}))\ ;\ T(x_{1'2'}) \geq 0\}$	$\{Z_{23}(T(x_{23}))\ ;\ T(x_{23}) \geq 0\}$
x_{ij} (mile)	0.82	0.61	0.03
Q_{ij} $[q_{ij}]$	-483.7981 223.1139 260.6843 3.0000 -5.0000 2.0000 245.6468 201.8504 -447.4971	-1000.0 500.0 500.0 0.1 -300.1 300.0 5.0 100.0 - 105.0	-1.0000 0.5000 0.5000 10.0000 -5.0000 8.0000 113.8891 120.2765 -234.1656

in the network have quite erratic structure. The transition rates q_{ij}, $i \neq j$ are not uniformly generated which leads to erratic behavior of CTMC $\{Z_{ij}(T(x_{ij})); t \geq 0\}$, $(i,j) \in E$. We made these ill-conditioned generator matrices purposefully to maximize the initial distribution impact on the expected travel time of each link.

Assuming independence, since the travel history of a vehicle does not affect the mean travel time of following link $(2,3)$, the mean travel time of link $(2,3)$, $m_1(x_{23})$, is a constant value. In contrast to the dependent case, as we already stated in Proposition 3.3, the value of $m_1(x_{23})$ varies depending on the preceding link travel history; either link $(1,2)$ or $(1',2')$. Thus, we experiment with how much the dependency of successive links would change the total expected shortest path and

whether or not it can change the $1 \rightarrow 3$ expected shortest path. The LP formulation of this problem is as follows: Define,

c_{jk}**:** The independent mean travel time of link (j, k).

$c_{i(jk)}$**:** The dependent mean travel time of link (j, k) emanating from node i or travelling from link (i, j).

X_{ij}**:** If link (i, j) is in the shortest path, then 1 otherwise 0.

$X_{i(jk)}$**:** If link (j, k) connected from link (i, j) is in the shortest path, then 1 otherwise 0.

If the links are independent, then the weight and the link binary variable have two indices. If the links are dependent, then they have three indices, indicating the preceding link history as we have shown in Section 3.4.

The independent expected $1 \rightarrow 3$ shortest path problem is formulated as,

$$\begin{aligned}
&\text{Minimize} \quad c_{12}X_{12} + c_{1'2'}X_{1'2'} + c_{23}X_{23} \\
&\text{Subject to} \\
&\quad X_{12} + X_{1'2'} = 1 \\
&\quad X_{23} - X_{12} - X_{1'2'} = 0 \\
&\quad X_{23} = 1 \\
&\quad X_{12}, X_{1'2'}, X_{23} \in \{0, 1\}
\end{aligned}$$

The dependent expected $1 \rightarrow 3$ shortest path problem is formulated as,

$$\begin{aligned}
&\text{Minimize} \quad c_{12}X_{12} + c_{1'2'}X_{1'2'} + c_{1(23)}X_{1(23)} + c_{1'(23)}X_{1'(23)} \\
&\text{Subject to} \\
&\quad X_{12} + X_{1'2'} = 1 \\
&\quad X_{1(23)} + X_{1'(23)} - X_{12} - X_{1'2'} = 0 \\
&\quad X_{1(23)} + X_{1'(23)} = 1 \\
&\quad X_{12}, X_{1'2'}, X_{1(23)}, X_{1'(23)} \in \{0, 1\}
\end{aligned}$$

Clearly depending on preceding link to $(2, 3)$, the link variable and weight of link $(2, 3)$ varies in the dependent case. Even with this $|N| = 3$, $|E| = 3$ graph, the number of decision variables is 4.

In this problem, we experiment with three cases of this expected shortest path problem; the independent case, the dependent of expected terminal distribution (dependent I) and the expected terminal state (dependent II) cases. Mainly, we focus on how the expected shortest path change based on different approaches and some of the important performance measures of link $(2, 3)$ as varying methods.

Table 4.5 The experiment result.

Variation \ Responses		Probability measures of link (2 ,3)		Expected travel time ($m_1(x_{23})$)	Expected total travel time of shortest path.
		Initial distribution	Marginal distribution at $m_1(x_{23})$		
Independent	**Path I**	[1 0 0]	[0.9396 0.0579 0.0025]	0.0742	**3.9345**
	Path II				3.9435
Dependent I	Path I	[0.0112 0.9806 0.0082]	[0.5756 0.4163 0.0081]	0.1139	3.9751
	Path II	[0.0037 0.2537 0.7426]	[0.6733 0.3201 0.0066]	0.0999	**3.9700**
Dependent II	Path I	[0 1 0]	[0.5923 0.4017 0.0061]	0.1610	4.0141
	Path II	[0 0 1]	[0.7591 0.2368 0.0041]	0.1159	**3.9860**
Asymptotic Measures	**Path I**		[0.8809 0.1178 0.0012]	0.0766	**3.1875**
	Path II				3.9485

As seen in Table 4.5, the shortest path changed by imposing the dependence from path I to path II and from the asymptotical perspective, path I turns out to be the shortest path again. Table 4.5 shows the various initial distributions of final link $(2, 3)$ and its corresponding marginal transient probability as well as the mean travel time of the link $m_1(x_{23})$. Next we analyze the behavioral aspects of CTMC governing link $(2, 3)$ with the marginal state transient probability to see how fast the marginal state transient probability converges to steady state.

As we already showed in an earlier section, the marginal state probability of the final link $(2,3)$ with initial state distribution can be found using Chapman-Kolmogorov equation as follows (Equations (4.1) and (4.2)),

$$\alpha(t) = [P\{Z_{23}(0)\}] \left[A \exp\{Dt\} A^{-1}\right] \tag{4.7}$$

Figure 4.5 shows the behavior of the marginal state probability of link $(2,3)$ dependent on preceding link $(1',2')$ with respect to various initial distribution due to methodologies: independent, dependent I (expected terminal distribution) and dependent II (expected terminal state). Considering the mean travel times of link $(2,3)$

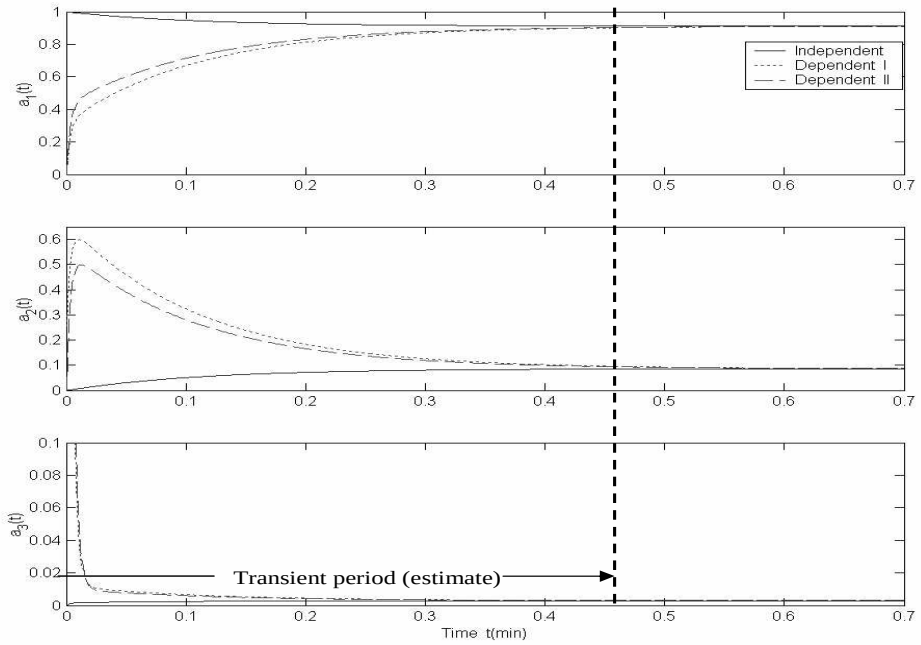

Figure 4.5 The behavior of $\alpha(t)$ of link $(2,3)$ from $(1',2')$ with respect to various z_0.

based on the dependence we clearly see each mean travel time of link $(2,3)$ lies before the termination of the transient period (approximately $0.48(min)$). Consequently, we confirm that the dependency impact lies in the transient period.

Next, along with Q matrix factor, the velocity function also turns out to enhance the impact of initial distribution when we use the exponential velocity instead of linear velocity function (Table 4.6) for link $(2, 3)$. Using the exponential velocity

Table 4.6 Mean travel time of link (2,3) with distinct velocity function.

Methodologies / Velocity Function	Independent	Dependent I		Dependent II		Asymptotic
		Path 1	Path 2	Path 1	Path 2	
Linear velocity case (V = 25 / i, i∈ S)	0.0742	0.1139	0.0999	0.1610	0.1159	0.0766
Exponential velocity case (V = 25 / exp(i), i ∈ S)	0.2036	0.2800	0.2590	0.2811	0.2517	0.2055

function model made a more dramatic difference in the mean travel time of link $(2, 3)$ between the independent case and the dependent case. Next, we present more formal numerical examples of our independent and dependent link methodologies.

4.2 Expected Shortest Path Problem I

In this section, we consider a numerical example in which most of the links contain an erratic CTMC structure so that the initial distribution impact persists. First, we find the independent expected shortest path in this network configuration and then we impose two distinct dependencies to find the expected shortest path: the dependence through the expected terminating state approach and dependence through the expected terminating distribution approach.

The numerical problem is the one introduced in Section 3.4 in the linear programming approach. The problem network and topology is shown in the Figure 4.6. There are length and traffic conditions for each link in Table 4.7. This network has 5 nodes and 8 links. The numbers in the generator matrix represent the range of the uniform distribution used to create the matrices. Thus, in the case of high density traffic II condition the second column of each row are generated from $200 \sim 300$ and the last column of each row is generated from $1 \sim 10$. The sample space of CTMC

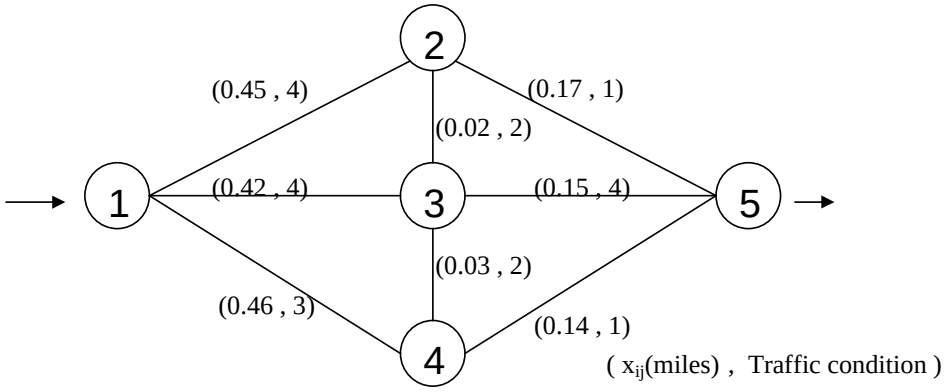

Figure 4.6 Experiment problem network topology.

Table 4.7 Experiment problem configuration.

Configuration / Factors	Configuration			
Generator matrix (Q) Random transition rate range	**High density traffic I (3)**	**High density traffic II (4)**	**Moderate density traffic (2)**	**Low density traffic (1)**
	$\begin{bmatrix} \\ 50 \sim 100 \\ \\ \end{bmatrix}$	$\begin{bmatrix} 200 & 200 & 1 \\ \sim & \sim & \sim \\ 300 & 300 & 10 \end{bmatrix}$	$\begin{bmatrix} 20 \sim 40 \\ 0 \sim 0.1 \\ 20 \sim 40 \end{bmatrix}$	$\begin{bmatrix} 0 \sim 10 \\ 0 \sim 0.1 \\ 0 \sim 10 \end{bmatrix}$

of each link and the velocity space are assumed to be homogeneous and each is given as,

$$S = \{1, 2, 3\}, \quad \nu = \{v_i : v_i = \frac{25}{\exp(i)}, i \in S\}.$$

First, we compute the independent expected shortest path in the network. We could use the methodology developed in Section 3.2 in which we use Dijkstra's algorithm to find the shortest path using the independent mean travel time of each links. However, for the purpose of comparison with dependent approaches where we use the K-shortest path heuristic, we decide to use one of the steps in the K-shortest path heuristic: the K shortest path algorithm with independent mean travel time of all

links. This provides us more information by ranking paths than Dijkstra's algorithm with which we only get the shortest path. In addition, the K shortest path algorithm encompasses the Dijkstra's algorithm in terms of finding the dominant shortest path set. Then we solve the dependent I and II with K-shortest path heuristic. We seek to observe the variation of K dominant paths' total travel time with respect to methodologies: independent, dependent I and dependent II approach. This observation is thought to give us an idea about the appropriate choice of K when using the K shortest path heuristic for dependent network problems, since the computational time is mainly determined by the K value. We also study performance of the K-shortest path heuristic in finding the optimal path with respect to behavioral aspects of CTMC: transient and asymptotic.

Finally, our objective is to find the expected shortest path from node 1 to 5 with respect to independent, dependent I and dependent II approach. Further, we experiment with different values of K for the dependent network cases.

4.2.1 Independent network links

We assume all links in the network are independent of each other and each has its own environment process unaffected by any adjacent environment. We use the methodology developed in Section 3.2. to find the expected shortest path along with the K shortest path algorithm introduced in step 1 in K-shortest path heuristic. In addition, we use the independent parametric total travel time distribution algorithm developed in Section 3.4.

First, we find the shortest path using the independent methodology and in order to compare the results of dependent cases, we use the K shortest path algorithm. Initially, we set $K = 4$ for this algorithm which means we compute the 4 shortest paths with the K shortest path algorithm.

The results are shown in Table 4.8. It turns out that path $1 \rightarrow 3 \rightarrow 5$ is the expected shortest path with expected total travel time of 5.8036 min. We need to

Table 4.8 Independent shortest path results.

Independent Approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 3 → 5** **(4.3198) (1.4839)**	**5.8036**
2nd shortest	**1 → 2 → 5** (3.4242) (2.4396)	**5.8638**
3rd shortest	**1 → 2 → 3 → 5** (3.4242) (1.3972) (1.4839)	**6.3053**
4th shortest	**1 → 4 → 5** (4.6079) (2.4115)	**7.0194**

note that the result in Table 4.8 is also the K dominant set in the K-shortest paths heuristic which is used to find the shortest *dependent* path among them in next two subsections.

Next we compute the parametric travel time distribution of this shortest path $1 \rightarrow 3 \rightarrow 5$ with both transient measures and asymptotic measures as follows (Table 4.9). The distribution of either transient normal and asymptotic normal travel time

Table 4.9 Normal distribution parameters of the path $1 \rightarrow 3 \rightarrow 5$.

Measures \ Link		(1,3)	(3,5)	Total path
Mean (min)	Transient	4.3198	1.4839	**5.8036**
	Asymptotic	4.3178	1.4818	**5.7995**
Variance (min)	Transient	0.00444	0.00126	**0.0057**
	Asymptotic	0.0100	0.0084	**0.0184**

of the shortest path is given in Figure 4.7. With the asymptotic perspective, we can expect larger variance than in the transient perspective which matches our result in Figure 4.7. In addition, assuming asymptotic steady-state CTMC of all links the mean travel time of each link becomes independent of the initial distribution which in this problem reduces the mean travel time of all links in path $1 \rightarrow 3 \rightarrow 5$ as

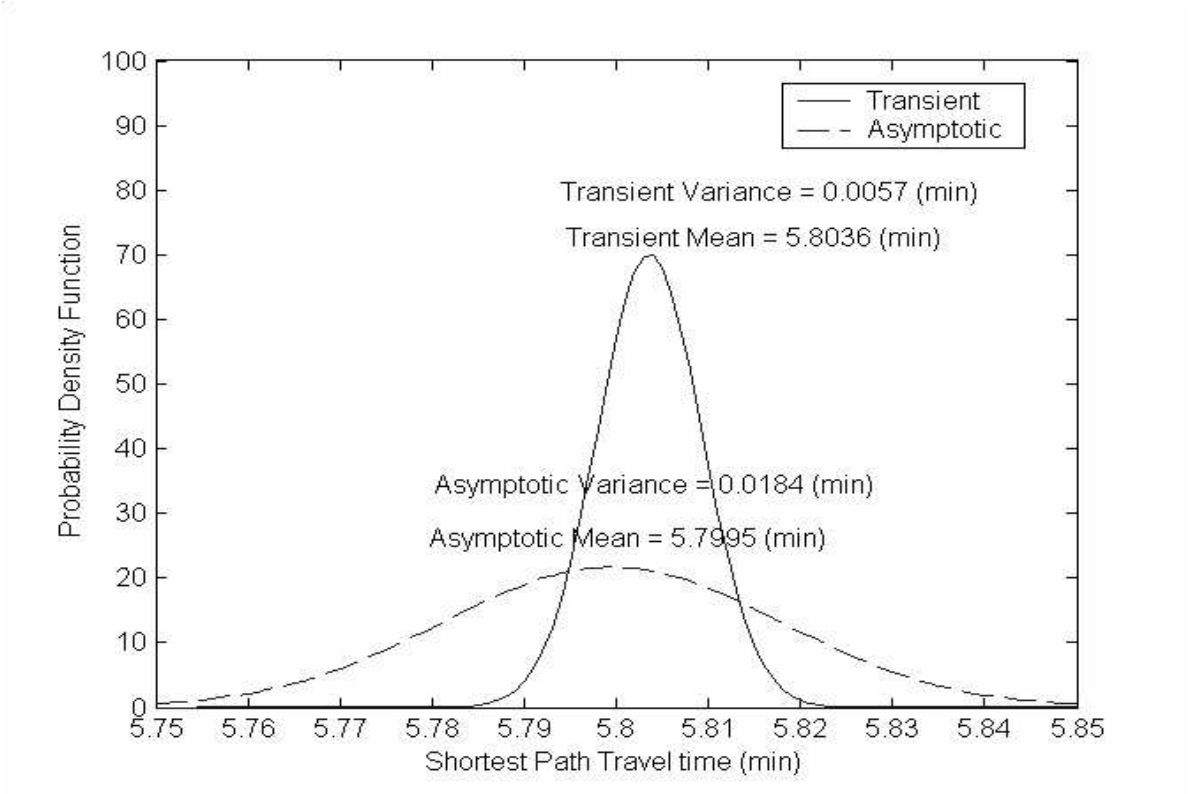

Figure 4.7 The independent shortest path travel time distribution.

compared to the transient state assumption. Both transient and asymptotic variance are relatively very small compared to mean and the coefficient variation of the transient and asymptotic normal distributions is 0.0130 and 0.0234, respectively, and coefficient variation of asymptotic normal distribution is bigger than the transient normal distribution in this problem. We can utilize this parameterized travel time distribution of the independent expected shortest path to gain useful information about this path travel time such as the probability that a vehicle can finish the trip on the path in less than some desired time.

Next, we impose the dependence on successive links in the network and find the dependent expected shortest path. First we seek the shortest path with the dependent I approach.

4.2.2 Dependent network links: Dependent I

We impose the dependence through the expected terminal distribution of preceding link (Dependent I approach). In our K-shortest path heuristic, we set K=4 and the step 1 result is shown above (Table 4.8). For comparison purposes, we show the dependent expected travel time of dominant independent shortest path set in Table 4.10. As we see in Table 4.10, imposing the dependence through expected

Table 4.10 Dependent I shortest paths.

Dependent I Approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 3 → 5** **(3.4242) (0.7707) (1.4833)**	**5.6782**
2nd shortest	**1 → 3 → 5** (4.3198) (1.4822)	**5.8020**
3rd shortest	**1 → 2 → 5** (3.4242) (2.3824)	**5.8066**
4th shortest	**1 → 4 → 5** (4.6079) (2.3490)	**6.9569**

terminal distribution causes a difference in the expected total travel time. However, overall expected travel time of the first $K = 4$ expected shortest paths tends to remain around $5 \sim 6$ minutes.

The expected shortest path turned out to be path $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$ with expected travel time of 5.6782 minutes which was the 3^{rd} expected shortest path in the independence case (Table 4.8). This is caused by the highly erratic structure of CTMC along most of the links.

However, we need to note that this result is theoretically not guaranteed to be the optimal solution because we only choose the shortest path among the dominant set of paths based on independent mean travel time. Namely, among the dominant paths in set $P = \{(1,3,5),(1,2,5),(1,2,3,5),(1,4,5)\}$ shown in the independent case above we recalculate the dependent mean travel time of each path and rank them (K-shortest path heuristic, Section 3.4). In order to find the optimal solution,

an enumeration algorithm should be used so that the mean travel time of all possible non-cyclic paths are computed. However, based on the analysis results in Section 4.1 where dependence impact tends to be negligible for non-erratic Q matrix, we can infer that the solution would be the actual optimal solution and it is not likely that the optimal path is contained in the dominated paths set $\bar{P}$ where $\bar{P} = E - P$.

Hence, in order to verify the above heuristic solution, we enumerate the rest of the dominated path set and compute the mean travel times and rank them. Table

Table 4.11 Dependent I dominated paths.

Dependent I	Path (Mean travel time)	Total mean travel time (min)
5th shortest	**1** → **4** → **3** → **5** (4.6079) (0.9975) (1.4835)	**7.0889**
6th shortest	**1** → **3** → **4** → **5** (4.3198) (0.7159) (2.4417)	**7.4774**
7th shortest	**1** → **2** → **3** → **4** → **5** (3.4242) (0.7707) (0.8517) (2.4368)	**7.4834**
8th shortest	**1** → **3** → **2** → **5** (4.3198) (0.7604) (2.4480)	**7.5281**
Longest	**1** → **4** → **3** → **2** → **5** (4.6079) (0.9975) (1.0176) (2.4451)	**9.0681**

4.11 agrees with our intuition that no other dominated path in the set $\bar{P}$ in terms of independence mean travel time turns out to be shorter than any dominated path in set P.

Next, we impose the dependence approach II on this problem to find the expected shortest path.

4.2.3 Dependent network links: Dependent II

We impose the dependence through the expected terminal state of the preceding link on the initial distribution of the following link CTMC. The result is given in Table 4.12.

The expected shortest path turns out to be $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$ with total mean travel time 5.4593 minutes. The path $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$ exactly matches that of

Table 4.12 Dependent II shortest paths.

Dependent II Approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1** → **2** → **3** → **5** **(3.4242)** **(0.5521)** **(1.4830)**	**5.4593**
2nd shortest	**1** → **3** → **5** (4.3198) (1.4830)	**5.8028**
3rd shortest	**1** → **2** → **5** (3.4242) (2.4526)	**5.8768**
4th shortest	**1** → **4** → **5** (4.6079) (2.4524)	**7.0603**

dependent I case above with a difference of required expected travel time 0.2489 minutes.

For the Dependent II approach, we enumerate the dominated path set $\bar{P}$, compute the mean travel time with Dependent II approach to verify the optimality of the solution, and rank them with dominated path set P. The result is given in the Table 4.13. Interestingly, the path $(1, 4, 5)$ which was selected as the 4^{th} dominant

Table 4.13 Dependent II dominated paths.

Dependent II Approach	Path (Mean travel time)	Total mean travel time (min)
4th shortest	**1** → **4** → **3** → **5** (4.6079) (0.5520) (1.4839)	**6.6438**
5th shortest	**1** → **2** → **3** → **4** → **5** (3.4242) (0.5521) (0.5520) (2.4524)	**6.9808**
6th shortest	**1** → **4** → **5** **(4.6079)** **(2.4524)**	**7.0603**
7th shortest	**1** → **3** → **4** → **5** (4.3198) (0.5520) (2.4524)	**7.3242**
8th shortest	**1** → **3** → **2** → **5** (4.3198) (0.5521) (2.4526)	**7.3245**
Longest	**1** → **4** → **3** → **2** → **5** (4.6079) (0.5520) (1.3972) (2.4396)	**8.9967**

path in Table 4.8 in terms of independent mean travel time turns out to be the 6^{th} shortest path following the paths $(1, 4, 3, 5)$ and $(1, 2, 3, 4, 5)$. However, the shortest

path $(1, 2, 3, 5)$ turns out to be the optimal path. If we are interested in finding the actual 4^{th} dependent shortest path, we should have at least set the parameter $K = 6$ for the K-shortest path heuristic.

Finally, in the Dependent II case, the shortest path found with the K-shortest path heuristic also turns out to be the actual optimal dependent shortest expected path. Through the two types of dependence results we can state that reflecting the reality (dependence) produces significant difference not only in the expected total travel time, but also the expected shortest path.

Thus far, we have used the transient distribution in imposing the dependence for subsequent links in the network as well as the transient mean travel time. Next, we present the asymptotic results of the independent, Dependent I and Dependent II expected shortest path in the following section.

4.2.4 Asymptotic network links

This section implements the method of asymptotic dependence developed in Section 3.3. There are three kinds of asymptotic expected shortest paths possible. First, we can be interested in the asymptotically expected shortest path where mean travel time of each link is computed based on the limiting behavior of the CTMC of the link in the network. In this case, since we focus on the limiting behavior of CTMC, the impact of the initial distribution is assumed to vanish. Consequently, asymptotic mean travel time becomes independent of preceding link. Otherwise, we can compute the asymptotic termination conditions of the preceding link: asymptotic expected terminating probability and asymptotic expected terminating state.

We use the same K-shortest path heuristic where instead of transient measurement we use asymptotic measurements. The asymptotic mean shortest paths result is given in Table 4.14. Compared to the independent shortest path (Table 4.8), overall the mean travel time down to the 4^{th} shortest path reduces within the range ± 0.1 minutes.

Table 4.14 Asymptotic shortest paths.

Asymptotic independent approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 3 → 5** **(3.4223) (0.8684) (1.4818)**	**5.7724**
2nd shortest	**1 → 3 → 5** (4.3178) (1.4818)	**5.7995**
3rd shortest	**1 → 2 → 5** (3.4223) (2.4481)	**5.8703**
4th shortest	**1 → 4 → 3 → 5** (4.6027) (0.8839) (1.4818)	**6.9684**

Next, Tables 4.15 and 4.16 show the asymptotic dependent expected shortest path.

Table 4.15 Asymptotic Dependent I shortest paths.

Asymptotic dependence I approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 3 → 5** **(3.4242 (0.7707) (1.4835)**	**5.6784**
2nd shortest	**1 → 3 → 5** (4.3198) (1.4822)	**5.8020**
3rd shortest	**1 → 2 → 5** (3.4242) (2.3824)	**5.8066**
4th shortest	**1 → 4 → 3 → 5** (4.6079) (0.9975) (1.4835)	**6.9569**

We need to note that these asymptotic dependent approaches use the asymptotic termination condition of the preceding link (steady-state) to impose the dependence for the initial distribution of the following link. We still computed the transient mean travel time as opposed to the asymptotic mean travel time (Table 4.14) while using only the asymptotic termination condition for the dependence models (expected terminal distribution or expected terminal state).

With the exception of the asymptotic mean travel time case, the shortest path of the asymptotic dependent links are identical to that of the transient dependent links (Tables 4.10 and 4.12).

Table 4.16 Asymptotic Dependent II shortest paths.

Asymptotic dependence II approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 3 → 5** **(3.4242) (0.5521) (1.4839)**	**5.4602**
2nd shortest	**1 → 3 → 5** (4.3198) (1.4830)	**5.8028**
3rd shortest	**1 → 2 → 5** (3.4242) (2.4526)	**5.8768**
4th shortest	**1 → 4 → 3 → 5** (4.6079) (0.5520) (1.4839)	**6.6438**

Next, we analyze the impact of the different dependence approaches on the shortest path and the K shortest path algorithm along with the K value analysis.

4.2.5 Analysis

As expected, the Dependent II approach of expected termination state caused more dramatic differences in mean travel time than Dependent I approach of expected termination distribution as shown in the Table 4.17. Particularly, the most

Table 4.17 Dependent expected travel time differences with independent case.

Dominant paths / Approaches	**1 → 3 → 5**	**1 → 2 → 5**	**1 → 2 → 3 → 5**	**1 → 4 → 5**
Dependence I Difference	-0.0016	-0.0572	-0.6271	-0.0625
Dependence II Difference	-0.0008	+0.013	-0.846	+0.0409

significant difference along the path $(1, 2, 3, 5)$ exists. In detail, this is mainly caused along the link $(2, 3)$ whose Q matrix is shown in Appendix D.

Next, we found that with $K = 4$ for the K-shortest path heuristic, we actually have *the optimal* dependent expected shortest path based on enumeration of all path travel times (Tables 4.10 and 4.11 for dependent I, Table 4.12 and 4.13 for dependent II). Since the rank of K dominant paths chosen based on independent measures

Table 4.18 Optimality and computational time of the heuristic with respect to varying K value and approaches.

Approaches / K value	Transient approach	Shortest path	Asymptotic approach	Shortest path
2	13.0690(sec)	Not optimal	9.1630(sec)	Optimal
4	18.0360(sec)	Optimal	10.8660(sec)	Optimal
6	23.7940(sec)	Optimal	18.3960(sec)	Optimal

change little when imposing the dependence, we can conjecture that we would have found the optimal dependent shortest path with a smaller value of K with less computational time. In addition, when computing each mean travel time (independent, Dependent I and Dependent II), we use either transient measures or asymptotic measures. The K shortest path heuristic with the asymptotic dependence approach is thought to solve the problem much faster than with the transient dependence approach. Thus, we analyze the performance of the heuristic with respect to the K value along with transient and asymptotic approaches. In this experiment, we seek the computational time of the heuristic with varying K value and two distinct types of link dependence: transient and asymptotic. Finally, we solve the dependent I approach problem again with varying K value and distinct CTMC behavioral approaches. This algorithm complexity and effectiveness comparison result is given in Table 4.18. Clearly, the computational time with asymptotic measures is less than that of the transient measures. The optimal path clearly changes according to the method we use to find the shortest expected path. The optimality of the shortest path found with varying K values is determined by comparing it to the true optimal shortest path found through all enumerations. Thus, this result suggests that for the network problem the K shortest path heuristic with asymptotic measures tends to need a smaller K value than with transient measures. Further, this result matches our insight that using asymptotic measures makes the expected shortest path less variable because of its independence of time.

4.3 Expected Shortest Path Problem II

In this section, we present another problem where the sample space of CTMC of each link is distinct across the network and the Q matrix structure is more moderate than previous problems. By moderate Q matrix structure, we mean each transition rate element q_{ij} remains within a certain comparably narrow range. In addition, the topology of the problem is bigger than the previous problem. Since we already analyzed various aspects of the expected shortest path problem in Section 4.2, in this section we state the problem and present the results directly.

The network topology and the problem configuration are as in Figure 4.8 and Table 4.19, respectively. We explicitly construct this problem according to the

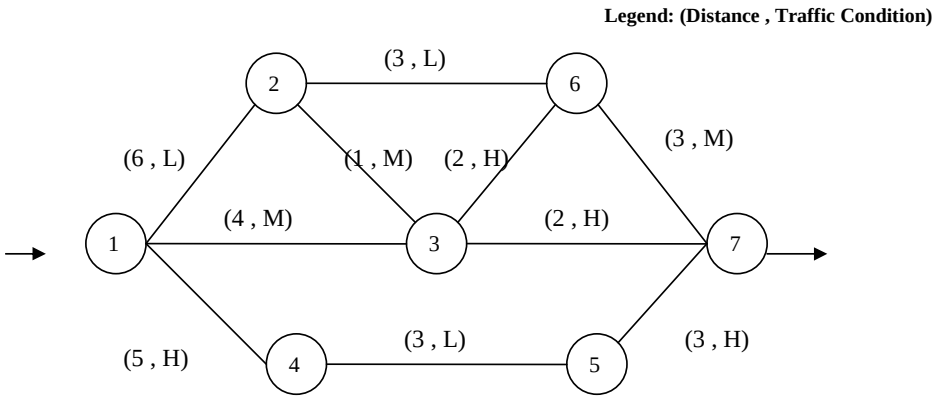

Figure 4.8 Experiment problem network topology.

stochastic network model developed in Section 3.1 where we introduced three kinds of traffic density (and corresponding Q matrices) and the sample space of CTMC: high density, moderate density and low density traffic link. However, obviously these are tentative categorizations and are not validated with the real world data. For each traffic density condition, the distinct Q matrix and sample space are specified in Table 4.19. Now, we run the K-shortest path algorithm to find the independent optimal K expected shortest paths and dependent K expected shortest paths. Based on the

Table 4.19 Experiment problem configuration.

Configuration / Factors	Configurations			
Velocity function ($V_{Z(T)}$) (mile/hour)	V = 65 / (Exp(Z(T)) , For ∀ Z(T) ∈ S			
Q matrix & Sample Space	Traffic condition	High density traffic **(H)**	Moderate density traffic **(M)**	Low density traffic **(L)**
	Q matrix random rate generation	[100 ~ 200]	[50 ~ 100]	[0 ~ 50]
	CTMC Sample Space (Z(T) ∈ S)	S = { 1, 2, 3, 4, 5, 6 ,7 }	S = { 1, 2, 3, 4, 5 }	S = { 1, 2, 3}

previous experiment (Figure 4.6, Table 4.7), we knew K-shortest path heuristic is comparably robust to find the dependent expected shortest path in terms of K value. Thus, we specified $K = 3$ for this problem which means that we seek down to the 3^{rd} expected shortest path.

The purpose of this experiment is first to consider the problem where the sample space is not identical across network links as it was assumed to be in the previous problem. That is, for a graph $G(N, E)$, there exist links (i, j),(j, k) such that

$$S_{ij} \neq S_{jk} \qquad (i, j), (j, k) \in E$$

where S_{gh} denotes the sample space of the CTMC of link $(g, h) \in E$. In addition, no Q matrices in the network are erratic as in the previous problem. The transition rates in each Q matrix comparably remain within stable ranges according to each traffic condition (Table 4.19).

Dependence of initial CTMC condition on the terminal condition of the preceding link's CTMC should be applied while we have to match two distinct successive

sample spaces accordingly. Terminal condition of the preceding link's environment process may no longer directly be interpreted as the initial condition of the next link's environment process whose sample space might be distinct. However, considering the fact that we still use homogeneous velocity functions across the network, we are easily able to match two distinct sample spaces. Details about how to impose the dependence across distinct sample spaces are given in Appendix C.

With $K = 3$, we run the K shortest path algorithm to find the K independent expected shortest path and run the K shortest paths heuristic to find the dependent expected shortest paths in the next section. The independent, dependent I and dependent II expected shortest path across distinct environment CTMC sample spaces in the network are found in Table 4.20. The independent expected shortest path

Table 4.20 Independent shortest paths.

Independent approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 6 → 7** **(32.8284) (14.9380) (23.3108)**	**71.0772**
2nd shortest	**1 → 3 → 7** (37.8604) (37.9644)	**75.8247**
3rd shortest	**1 → 2 → 3 → 7** (32.8284) (8.1168) (37.9644)	**78.9095**

turned out to be $1 \to 2 \to 6 \to 7$ with mean travel time 71.0772 minutes and variance 1.2474 minutes. It is interesting to note that the path $(1, 2, 6, 7)$ is the longest path from node 1 to node 7 in terms of only distance, $6+3+3 = 12$ miles. However, in terms of the traffic density level, the path $(1, 2, 6, 7)$ has the overall lowest density level, $\{L, L, M\}$. Hence, this result reflects the reality that traffic conditions play an important role in travel time on certain paths.

The parametric approximated travel time along the path $(1, 2, 6, 7)$ is given in the Figure 4.9.

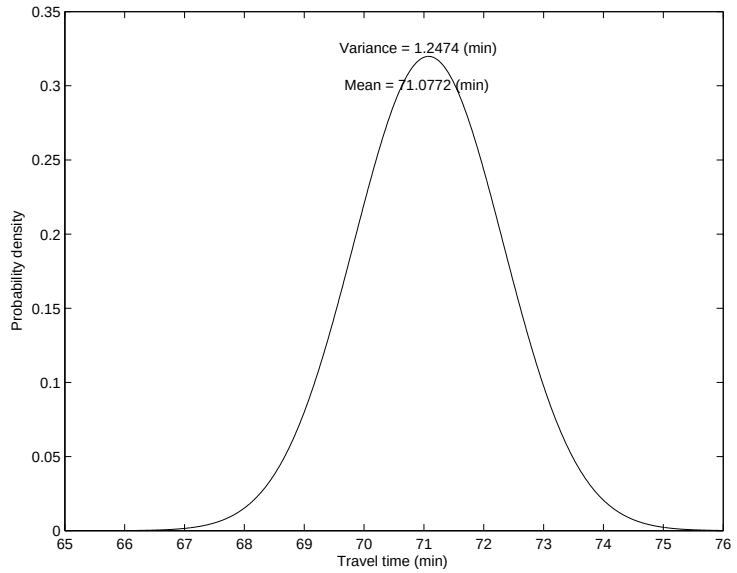

Figure 4.9 The independent shortest path travel time distribution.

Next, the Dependent I and II expected shortest paths are given in Table 4.21 and Table 4.22, respectively. In fact, the variation of expected travel time with

Table 4.21 Dependent I shortest paths.

Dependent I approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 6 → 7** **(32.8284) (14.9284) (23.3065)**	**71.0633**
2nd shortest	**1 → 3 → 7** (37.8604) (37.9625)	**75.8229**
3rd shortest	**1 → 2 → 3 → 7** (32.8284) (8.1128) (37.9622)	**78.9034**

respect to dependence impact on successive link travel time turned out to be negligible in this problem. The expected shortest path does not change as we impose the various dependency models in computing the expected travel time of successive links. The path $(1, 2, 6, 7)$ still remains as the expected shortest path with dependence approaches. Moreover, the difference between independence expected travel

Table 4.22 Dependent II shortest paths.

Dependent II approach	Path (Mean travel time)	Total mean travel time (min)
1st shortest	**1 → 2 → 6 → 7** **(32.8284) (14.9337) (23.3077)**	**71.0698**
2nd shortest	**1 → 3 → 7** (37.8604) (37.9635)	**75.8238**
3rd shortest	**1 → 2 → 3 → 7** (32.8284) (8.1131) (37.9635)	**78.9049**

time and dependence travel time falls within ±0.01 minutes for the top 3 expected shortest paths.

In the configuration of this problem (Table 4.19), the Q matrix elements are generated to be moderate compared to the way it was generated in the problem in section 4.2 (Table 4.7). The structure of the Q matrices of each link can be compared in Appendix D. The impact of variation of initial distribution according to the preceding link tends to vanish so fast that the dependence on the preceding link travel history does not make significant differences in expected travel time on the following link.

5. Conclusions

In this research, we considered stochastic and dynamic transportation network problems. First, we built a stochastic network model and developed methodologies to solve the expected shortest (least-time) path in the network. In the expected shortest path problem, we had to capture the dependent nature of successive links in the network. Further, the shortest path total travel time distribution was sought in the stochastic network.

For modelling of a stochastic network, the environment process methodology was extended for individual link to an entire stochastic network reflecting various traffic conditions in the real-world by utilizing the three factors of the environment process: initial distribution, velocity function and Q matrix. For the expected shortest path problem, we first developed an algorithm for the independent network using a deterministic shortest path algorithm. Further, by capturing the interaction between the termination condition of the preceding link and the initial condition of subsequent link, we invented two kinds of dependence concepts for successive links travelling: expected terminal distribution and expected terminal state. Because of the dependence along the successive travel time on links, the Dijkstra type greedy algorithm may not provide the optimal solution. Thus, we developed the K-shortest path heuristic inspired by Fu and Rillet [20] to find and compute the dependent expected shortest path. Further, we introduced a possible asymptotic approach for both independent and dependent expected shortest path methodologies. Finally, we developed an algorithm to find the total travel time distribution of the shortest path with an approximated parametric distribution, and found that the expected shortest path is also a stochastically shortest path when employing the normality assumption for parametric distributions.

We provided three types of algorithms to find the expected shortest path in a stochastic network: independent, dependent I and dependent II expected short-

est path algorithms. For each of these approaches, we also considered the asymptotic behavioral aspect of the underlying CTMC of the environment process. These methodologies were implemented through two numerical example problems along with two other numerical examples executed for dependence analysis purposes.

Our model of expected shortest path and travel time distribution is a unique methodology for the stochastic network model built with the environment process. Since the expected shortest path along with its travel time distribution is found through our methodologies for stochastic networks, this path information along with its distribution is quite significant for transportation networks. This is considered to be a significant improvement for shortest path problems in a stochastic and dynamic network in that with velocity function for the entire network and Q matrix estimation for individual link, we can find the expected shortest path while capturing the dependence nature and its travel time distribution.

In the real world application, our model only requires us to collect data for the transition rates along each link. Considering the outputs of our model: the various expected shortest paths and the travel time distribution, the level of effort required to use our methodology is moderate. In military applications, one of the valuable applications would be the case where the velocity is interpreted as a rate of work completion such as the case of a project management network. Then, we might want to find the expected longest path (the critical path) which can easily be found with slight modifications to our algorithms. With a small effort of data collection and modifying the algorithm, we can easily find either independent or dependent expected longest path along with the parametric probability distribution of expected longest path.

However, our analysis of the model might still need to be verified and generalized through statistical experiments. Further, we need to use our methodologies with real-world data to validate and verify our analysis in this thesis. In addition to generalization and validation, for the methodology improvement we need to note

that dependence did not make a significant difference unless Q matrix structure of subsequent link's CTMC is erratic or semi-absorbing. This is caused by our fundamental concept to capture the dependence of initial condition of subsequent link's CTMC on the terminal condition of preceding CTMC. Initial condition impact tends to vanish as time goes on which also matches what we might find in the real world. However, we do not capture the possible interaction of two successive link's CTMC such as Q matrix interaction. As opposed to our dependence where each subsequent link depends on preceding link, in the real world the preceding link might be affected significantly by the following link, for instance a traffic congestion caused from downstream links. Future work might involve these possible mutual interactions between successive stochastic links. Furthermore, our parametric approximated probability distribution of independent expected shortest path travel time demands some restrictive assumptions while making a simplified parametric distribution. We might need to find a more accurate distribution reflecting the dependence impact on total travel time distribution. Thus, it is worthwhile to study the dependent travel time distribution in a stochastic and dynamic network.

Appendix A. Algorithm Codes

A.1 Hierarchy of MATLAB Functions

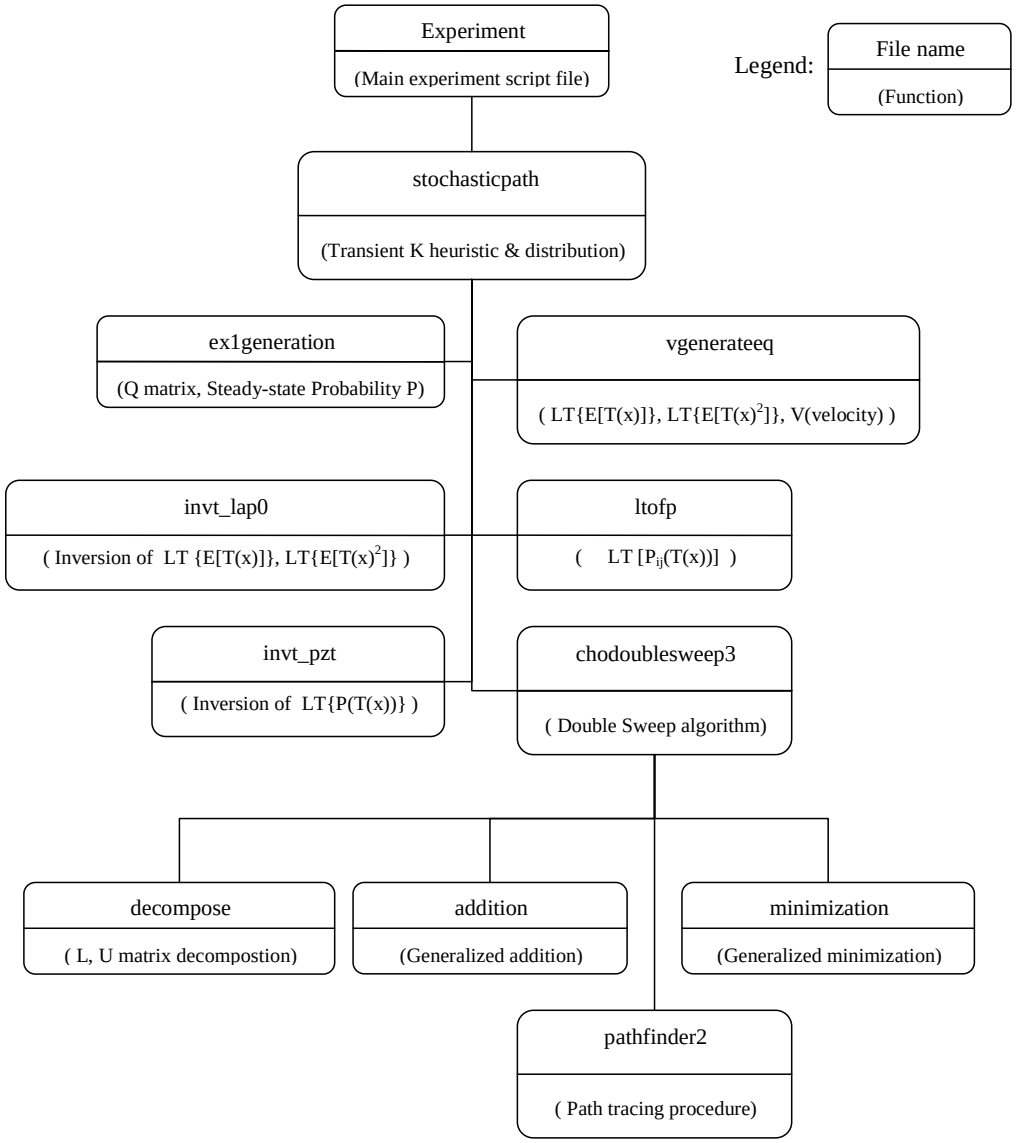

Figure A.1 Hierarchy of codes for transient methodologies.

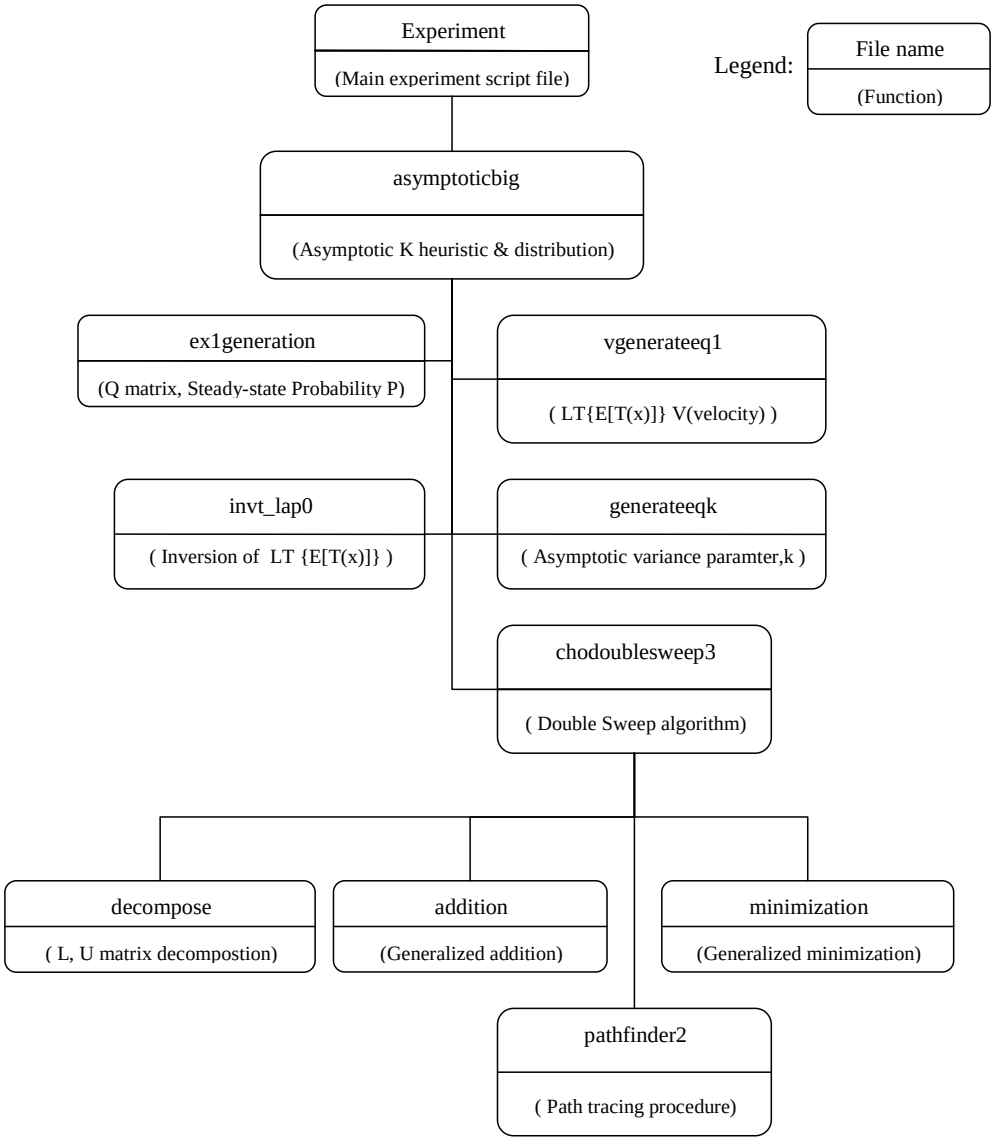

Figure A.2 Hierarchy of codes for asymptotic methodologies.

A.2 Transient Stochastic K-shortest path heuristic

```
**************************************************************************
 Function program Matlab code: stochasticpath
 This function program code is to compute and find the expected
 K-independent, expected dependent1,2 shortest paths in a given network.
 This algorithm provides ALL useful measures of the stochastic
 network;
 steady state of probability, transient & asymptotic mean travel time and
 all K shorest expected paths information as well as the distribution of
 the expected shortest path with both transient and asymptotic parameters.

 Algorithm procedure is following.
 Step 0 : Initialization & independent mean travel time of each link
 Step 1 : Double sweep algorithm to find K shortest paths.
 Step 2 : Computing the distribution of independent shortest path.
 Step 3 : Computing dependent mean travel time of all shortest paths.
 Step 4 : Ranking K shortest paths & printing results

 Input arguments
  A = distance matrix where Aij=if i and j are adjacent,distance between
      node i and j otherwise, Aij=inf
  D = Traffic condition matrix where Dij= traffic condition of link (i,j).
  st = Number  of state of the CTMC across the network.
  kth = K paramter value of K shortest path algorithm, down to kth shortest
        path.

 Output arguements
  Q = Q matrix cell array where Qij= Q matrix of CTMC of link(i,j)
  V = Velocity diagonal matrix where Vii= velocity of vehicle
      at state i in CTMC.
  ASYM = Asymptotic mean travel time matrix where ASYM(i,j)=asymptotic
         mean travel time of link (i,j)
  INM,DEM,DEM2 = Independent, dependent I and dependent II mean travel
                 time matrix where INM(i,j),DEM(i,j),DEM2(i,j)=mean travel
                 time of each link (i,j).
  EPT = Expected proability vector matrix where EPT(i,j)=the value of
        link(i,j):Dependent I
  EST = Expected state proability vector matrix where EST(i,j) =the value
        of link (i,j):Dependent II.
  PST = Steady state probability cell matrix where PST{i,j}= steady state
        probability vector of link (i,j).
  Dep1dis,Dep2dis,Dep1tot = Distance information storages for K paths of
```

```
                              dependentI and dependent II
  fD = Shortest paths value matrix whose element fD(ij)is a vector of
          k shortest paths from node i to j.
  a  = Mean of independent expected shortest path travel time
       distribution.
  b  = Variance of independent expected shortest path travel time
       distribution

  Kpath = K shortest path cell array where Kpath{i}= ith shortest path
          nodes storage

 Associated sub algorithm files
  -ex1generation.m file: Generation of Q matrix, steady state probability P.
  -vgenerateeq.m file: Transform of LT{E{T(x)},LT{E[T(x)]^2}.
  -ltofp.m file: Tranform of LT[P(T(x))].
  -invt_lap0.m file: Inversion of LT{E{T(x)},LT{E[T(x)]^2}.
  -invt_pzt.m file: Inversion of LT[P(T(x))].
  -chodoublesweep3.m file: Core algorithm file( K shortest paths).

 Author: Capt. Jae Il Cho
         Date  : 11/24/02
 Last Revision: 01/24/03
*************************************************************************

function
[Q,V,ASYM,INM,DEM,DEM2,EPT,EST,Z0,PST,Kpath,dis,Dep1dis,Dep2dis,Dep1tot,
          Dep2tot,fD,a,b]=stochasticpath(A,D,st,kth)

%Step 0: Initialization & independent mean travel time of each link
n=size(A,2); Q=cell([n,n]); ASYM=zeros(n,n); INM=zeros(n,n);
VARI=zeros(n,n); DEM=zeros(n,n); DEM2=zeros(n,n); EPT=cell([n,n]);
EST=cell([n,n]); Z0=cell([n,n]); PST=cell([n,n]);


tic; % Computation time intiated
%Generation of Q matrix, all initial measures;Mean,Steadystate probability etc.
for i=1:n
  for j=1:n
    if A(i,j)~=inf & i < j

       [Q1,P]=ex1generation(D(i,j),st)
       %[Q1,P]=ex2generation(D(i,j),st)
```

```
        z0=zeros(1,st);
        z0(st)=1;

        [LTm1,LTm2,V]=vgenerateeq(st,Q1,z0);%Find the LT{E{T(x)},LT{E[T(x)]^2}.
        m=invt_lap0(A(i,j),LTm1);           %Find the E{T(x)}.
        m2=invt_lap0(A(i,j),LTm2);          %Find the E[T(x)]^2.
        v=m2-m^2;                           %Var[T(x)]

        %Independent transient mean & variance
        INM(i,j)=m
        INM(j,i)=m
        VARI(i,j)=v
        VARI(j,i)=v

        % Asymptotic mean time.
        ASYM(i,j)=A(i,j)/sum(P*V);
        ASYM(j,i)=ASYM(i,j);

        %steady-state distribution & Q matrix.
        PST{i,j}=P;
        PST{j,i}=P;
        Q{i,j}=Q1
        Q{j,i}=Q1;

        %For only initial arcs emanating from source node.
        if i==1
         Z0{i,j}=z0;
         Z0{j,i}=Z0(i,j);

         %Expected probability at E[T(x)]:dependent I
         Ltp=ltofp(Q1);          %Generating LT[P[Z(t)}]
         Pt=invt_pzt(m,Ltp);     %Finding P(m)
         EPT{i,j}=z0*Pt;

         %Expected state at E[T(x)]:dependent II
         EST{i,j}=zeros(1,st);
         est=round(EPT{i,j}*[1:st]');
         EST{i,j}(1,est)=1;
        end
      end
    end
end
```

```
%Step 1: Double sweep algorithm to find K shortest paths.
%Modifying the INM matrix to fit the input form of chodoublesweep3.m file.
 for i=1:n
    for j=1:n
     if INM(i,j)==0
        INM(i,j)=inf;
    end
 end
end

%dijkstra(INM);
[L,U,fD,Kpath,dis,tot]=chodoublesweep3(INM,kth);



%Step 2: Computing the distribution of independent shortest path.
distrimean=tot(1); distrivari=0; for i=1:size(Kpath{1},2)-1
distrivari=distrivari+VARI(Kpath{1}(i),Kpath{1}(i+1)); end



%Step 3 : Computing dependent mean travel time of all shortest paths.
%Computing of expected dependent travel time of each selected
%shortest path in K paths.
Dep1dis=cell([1,kth]); Dep2dis=cell([1,kth]);
Dep1tot=zeros(1,kth); Dep2tot=zeros(1,kth); for i=1:kth
    %Starting link mean travel time.
    Dep1dis{i}(1)=INM(Kpath{i}(1),Kpath{i}(2))
    Dep2dis{i}(1)=INM(Kpath{i}(1),Kpath{i}(2))
    for j=2:size(Kpath{i},2)-1

     g=Kpath{i}(j)
     h=Kpath{i}(j+1)


    %Expected probability at E[T(x)]:dependent I
    z0=EPT{Kpath{i}(j-1),Kpath{i}(j)}; %EPT: Expected terminal probability
    [LTe]=vgenerateeq(st,Q{g,h},z0);
    DEM(g,h)=invt_lap0(A(g,h),LTe); %Dependent I mean travel time of link (g,h)
    Ltp=ltofp(Q{g,h});          %Generating LT[P[Z(t)}]
```

```
    Pt=invt_pzt(DEM(g,h),Ltp);  %Finding P(t) at E[T(x)]
    EPT{g,h}=z0*Pt;


    %Expected state at E[T(x)]:dependent II
    z02=EST{Kpath{i}(j-1),Kpath{i}(j)}; %EST : Expected terminal state
    [LTe]=vgenerateeq(st,Q{g,h},z02);
    DEM2(g,h)=invt_lap0(A(g,h),LTe);
    EST{g,h}=zeros(1,st);
    est=round(EPT{g,h}*[1:st]');
    EST{g,h}(1,est)=1;


    Dep1dis{i}(j)=DEM(g,h);
    Dep2dis{i}(j)=DEM2(g,h);

   end
   Dep1tot(i)=sum(Dep1dis{i});
   Dep2tot(i)=sum(Dep2dis{i});

 end

%Step 4 : Ranking K shortest paths & printing results
%Sort and keep indices of rank.
  [sorted1,index1]=sort(Dep1tot);
  [sorted2,index2]=sort(Dep2tot);

  t=toc;

 %Printing all necessary results !
 fprintf('Independent Case!')
 for i=1:kth
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',Kpath{i})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,dis{i})
 fprintf('Independent total travel time :total: %3.4f\n',tot(i))
end

 disp('')
 disp('')
fprintf('Dependent I Case!')
```

```
 for i=1:size(index1,2)
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',Kpath{index1(i)})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,Dep1dis{index1(i)})
 fprintf('Dependent I total travel time : %3.4f\n',Dep1tot(index1(i)))
 end
 disp('')
 disp('')
fprintf('Dependent II Case!') for i=1:size(index2,2)
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',Kpath{index2(i)})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,Dep2dis{index2(i)})
 fprintf('Dependent II total travel time : %3.4f\n',Dep2tot(index2(i)))
 end

fprintf('The independent expected shortest path distribution\n')
distrimean distrivari a=distrimean; b=distrivari;
 fprintf('Computational time')
t
```

A.3 Generation of Q matrices in Problem in Section 4.2

```
*************************************************************************
 Function program Matlab code: ex1generation.m
 This function code is to generate various somewhat erratic Q matrix based on
 traffic condition number associated with each link(Refer to Chapter 4.2 problem
 configuration).  For real application, this Q matrix should be generated
 based on tranistion rate estimate with collected data.  Thus, this file
 is only for the experiment problem in Chapter 4.2.

 Input arguments
  a = Traffic condition number(Refer to Chapter 4.2 problem configuration.
  k = The number of state.

 Output arguements
  Q = Q matrix generated.
  P = Steady state probability distribution vector associated above Q
      matrix.

 Author: Capt. Jae Il Cho
         Date  : 11/24/02
 Last Revision: 01/24/03
*************************************************************************
%Only associated with  experiment4-2.
function [Q,P] = ex1generation(a,k) A=zeros(k,k); if a==1
    b=0;
    c=10;
    d=0;
elseif a==2
    b=20;
    c=40;
    d=0;
elseif a==3
    b=50;
    c=100;
    d=0
else
   b=200;
   c=300;
   d=1;
   e=10;
end for i=1:k
    for j=1:k
```

```
     A(i,j)=unifrnd(b,c);% Generation of random variates
    end
 end

if a==2
 for i=1:k
     %A(1,i)=unifrnd(300,500);
     A(2,i)=unifrnd(0,0.1);
     A(k,i)=unifrnd(0,0.1);
 end
end if a==1
 for i=1:k
     %A(1,i)=unifrnd(50,70);
     A(2,i)=unifrnd(0,0.1);

  end
end if d~=0
  for i=1:k
    A(i,k)=unifrnd(d,e);
  end
end

for i=1:k
   A(i,i)=0;%Making diagonal empty
end

Ai=[]; Ai=sum(A,2); for i=1:k
    %Substituting negative of sum of offdiagonal elements into diagonal.
    A(i,i)=-Ai(i,1);
end
p=zeros(1,k);% Generating P
Q1=zeros(k,k);%Generating Q
e=ones(k,1);%Generating e
Q1=[A(1:k,1:k-1),e];% Generating Q~ : all Q elements but last all 1 column
b=zeros(1,k);
b1=[b(1,1:k-1),1];% Generating right hand side
p=b1*inv(Q1);%Solving linear equation for steady state distribution.
Q=A; P=p;
```

A.4 Laplace Transform of Lower Moments

```
*************************************************************************
 Function  Matlab code:vgenerateeq.m
 This function program code is to generate the diagonal velocity matrix
 and find the LT{E[T(x)]}, LT{E[T(x)^2]}.  We can generate the velocity
 matrix in accordance with various possible velocity function. Here we
 set two functions; linear and exponential function(exponential is
 default).

 Input arguments
  a = The number of state in sample space of CTMC
  Q = Q matrix of CTMC
  z0 = Initial probability distribution vector of CTMC

 Output arguements
  LTmeq = LT{E[T(x)]}
  LTm2eq= LT{E[T(x)^2]}
  V = Velocity diagonal matrix where Vii= velocity of vehicle
      at state i in CTMC.

 Author: Capt. Jae Il Cho
         Date  : 11/24/02
 Last Revision: 01/24/03
*************************************************************************

function [LTmeq,LTm2eq,V]=vgenerateeq(a,Q,z0) syms s v=[]; for
  i=1:a
  %v(i)=25/(i*60); % Linear velocity function
  v(i)=25/(60*exp(i)); % Exponential velocity function
end V=diag([v]); e1=ones(a,1);

D=s*V-Q; if nargout < 3
LTmeq=(1/s)*z0*inv(D)*e1; % LT{m(t(x))}
else if nargout ==3
LTmeq=(1/s)*z0*inv(D)*e1; % LT{m(t(x))}
LTm2eq=(1/s)*2*z0*(inv(D)*inv(D))*e1; else disp('Number of output
argument is wrong') end end
```

A.5 Numerical Inversion of Lower Moment Transforms

```
************************************************************************
 Function program Matlab code: invt_lap0.m
 The purpose of this MATLAB program is to inverse the LT of E[T(x)]
 to E[T(x)]  numerically.

 Input arguments
   - b : LT{E[T(x)]}
   - t : Length of arc x

 Output arguments
   - f1 : E[T(x)] where length of x is given an input(t)

        Author: Dr. Kharoufeh
        Date  : 11/24/02
 Last Revision: 01/24/03(Capt Cho)
************************************************************************

function f1 = invt_lap0(t,b) syms s x y

eq=b; s=x+y*i;

rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;

for k=0:m
   d=nchoosek(m,k);
   c=[c d];
end
for t = t; %50.0; %t=0.5:0.5:20.0
   tx = t; %[tx t];
   ntr=15;
   u=exp(A/2)/t;
   x=A/(2*t);
   h=pi/t;
   su=zeros(m+2);
   y=0;  %!
   s=x+y*i;
   sm=eval(eq)/2;% Modified ! .
   for k=1:ntr
        x=A/(2*t);
      y=k*h;
      s=x+y*i;
```

```
      sm=sm+((-1)^k)*real(eval(eq)); % !
   end
   su(1)=sm;
   for k=1:12
      n=ntr+k;
       x=A/(2*t);
      y=n*h;
      s=x+y*i;
      su(k+1)=su(k)+((-1)^n)*real(eval(eq));  %!
   end
   av1=0; av2=0;
   for k=1:12
      av1=av1+c(k)*su(k);
      av2=av2+c(k)*su(k+1);
   end
   f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end
```

A.6 Laplace Transform of CTMC Marginal Probabilities

```
%************************************************************************
% Function  Matlab code:ltofp.m
% This function program code is to find the Laplace transform of transient
% conditional state probability, LT[Pij(t)].  This function program code is
% to find the Laplace transform of transient conditional state probability,
% LT[Pij(t)].  This conditional state probability is used to compute the
% marginal transient state probability at mean travel time
% (terminal probability).

% Input arguments
%  Q = Q matrix of CTMC

% Output arguements
%  LTp = LT[Pij(t)]

% Author: Capt. Jae Il Cho
%         Date  : 11/24/02
% Last Revision: 01/24/03
%************************************************************************


function [LTp]=ltofp(Q)% Q=Qmatrix, b=z0 intial distribution vector



syms s
c=size(Q,1); %Number of rows of Q matrix
I=eye(c); %Generating identity matrix based on number of rows of Q matrix
D=s*I-Q; %Generating SI matrix and minus Q matrix.

LTp=inv(D); % LT of [P{Z(t)=i|Z(0)=j}] for all (i,j) in sample space.
```

A.7 Numerical Inversion of CTMC Marginal Probabilities

```
%************************************************************************
 Function  Matlab code: invt_pzt.m
 This function program code is to inverse  the Laplace transform of
 transient conditional state probability, LT[Pij(t)].

 Input arguments
  -Ltp = LT[Pij(t)]
  -t  = Time (mean travel time for terminal probability).
 Output arguements
  -f2 = Conditional probability matrix [Pij(t)].

 Author: Capt. Jae Il Cho
         Date  : 11/24/02
 Last Revision: 01/24/03
%************************************************************************

%When input argument Ltp is matrix:default.
function f2 = invt_pzt(t,Ltp)

n1=size(Ltp,2); %number of columns in b.
Pzt=[];

for f=1:n1 %For every element in p array find the P(z(t)).
    for g=1:n1
        b=Ltp(f,g);

syms s x y

eq=b; s=x+y*i;

rho=0.8; qx=[0.8]; tx=[0]; m=11; c=[]; ga=8; A=ga*log(10); mm=2^m;

for k=0:m
   d=nchoosek(m,k);
   c=[c d];
end
for t = t; %50.0; %t=0.5:0.5:20.0
   tx = t; %[tx t];
   ntr=15;
   u=exp(A/2)/t;
   x=A/(2*t);
```

```
   h=pi/t;
   su=zeros(m+2);
   y=0;  %!
   s=x+y*i;
   sm=eval(eq)/2;% Modified !Calling generateeq m. which contain LT(m(t)).
   for k=1:ntr
        x=A/(2*t);
      y=k*h;
      s=x+y*i;
      sm=sm+((-1)^k)*real(eval(eq)); % !
   end
   su(1)=sm;
   for k=1:12
      n=ntr+k;
       x=A/(2*t);
      y=n*h;
      s=x+y*i;
      su(k+1)=su(k)+((-1)^n)*real(eval(eq));  %!
   end
   av1=0; av2=0;
   for k=1:12
      av1=av1+c(k)*su(k);
      av2=av2+c(k)*su(k+1);
   end
   f1 = u*av1/mm; f2=u*av2/mm; qx=[qx f2];
end

Pzt(f,g)=f1;

end
end

f2=Pzt; %f2 = matrix of p(t).
```

A.8 Double Sweep Algorithm

```
*************************************************************************
 Function program Matlab code: chodoublesweep3
 The purpose of this MATLAB program is to compute the K shortest paths
 for all distance and find the source to destination K shortest paths.
 This is based on 'double sweep algorithm' introduced by Shirer
 in 'On algorithms for finding the k shortest paths in a network'
 [Networks9],1979 p195~p214 and slightly modified for cycle path
 reduction and path tracing procedures.

 Algorithm procedure is following.
 Step 0 : Initialization
 Step 1 : Double sweep algorithm
 Step 3 : Path tracking algorithm

 Input arguments
   - C : Distance matrix whose element Cij is the independent mean
         travel time between two node  Cij=inf for i=j.
   - k : Required k shortest paths value,k.

 Output arguments
   - L: Lower triangular portions of fD matrix
   - U: Upper trigngular portions of fD matrix
   - fD: shortest paths value matrix whose element fD(ij)is a vector
         of  k shortest paths from node i to j.
   - Path: Path list cell array whose row is a j path nodes list.
   - distant: cell array whose row is a successive distance between
              two successive nodes.
   - tot:   Total distance list from 1st to kst shortest path from
            source to sink node.

 Associated sub algorithm files.
   -minimization.m: Generalized minimization set operation with same
                    same dimensional two vectors.
   -addition.m: Generalized addition set operation with same dimensional
                two vectors.
   -decompose.m: Decomposition of estimates matrix into L and U matrix
   -pathfinder.m: Path tracing algorithm for k shortest path values

        Author: Capt. Jae Il Cho
        Date  : 11/24/02
```

```
 Last Revision: 01/24/03

**************************************************************************

Note! the order of element in distant[] is opposite of that of
path[]. Only difference with chodouble sweep are the lines around
188 Difference with chodoublesweep2 is whether sweeping the
diagonal element or not(here we skip the diagonal element to
minimize the cycle risk).


function [L,U,fD,Path,distant,tot]=chodoublesweep3(C,k)


%Step 0: Initialization

n=size(C,2);
%Consturcting k paths matrix [Dij] between node i,j where
%Dij= vector of k shortest paths between node i,j.
D=cell(n,n); for i=1:n
  for j=1:n
   if i==j
     D{i,j}(1)=0;
   else if C(i,j) ~= 0
     D{i,j}(1)=C(i,j);
   else
     D{i,j}(1)=inf;
   end
  end
    for h=2:k
     D{i,j}(h)=inf;
    end
  end
end

D0=D;


%Constructing semi-upper and lower triangular matrix of D.
[L,U]=decompose(D);
```

```
%Step 1 : double sweep algorithm

v=1; iter=1; BackD=D;
%As long as there is any differecne in two successive estimates.
while v~=0

 %Backward estimates sweep.
different=zeros(n,n);
for i=1:n  %Row
  D{i,n}=D{i,n};
 for j=n-1:-1:1

   if i~= j          %Skip the diagonal element in D.
   added=cell(1,n);  %Number of addition
   for h=1:n
    added{h}=addition(D{i,h},L{h,j});
   end
   minimized=added{1};

   for o=1:n-1
   minimized=minimization(minimized,added{o+1});
   end
   %Backward new replacement of element.
   D{i,j}=minimization(minimized,D{i,j});
   end

 %Termination condition test.
   difference=setdiff(D{i,j},BackD{i,j});
   different(i,j)=~isempty(difference);

 end
end
% Number of nonzero elements in matrix different(termination condition)
v=nnz(different);

BackD=D; celldisp(D)


%Forward estimates sweep.
for i=1:n  %Row
```

```
  D{i,1}=D{i,1};
 for j=2:n
   if i~= j
   added=cell(1,n);  %Number of addition
   for h=1:n
    added{h}=addition(D{i,h},U{h,j});
   end
   minimized=added{1};

   for o=1:n-1
   minimized=minimization(minimized,added{o+1});
   end
   D{i,j}=minimization(minimized,D{i,j});
   %Checking the difference.

   end
 end
end


iter=iter+1;
end

fD=D;


%Step 2: Path trakcing algorithm


%Tracking k shortest paths from source to sink.




%Initialization of storage of information.
cho_Adjacent=cell(n,1); P=cell(1,k); Path=cell(1,k);
dis=cell(1,k); distant=cell(1,k);

%Making adjacent matrix where column vector is the adjacent node list.
for i=1:n
   cho_Adjacent{i}=find(C(:,i)~=0 & C(:,i)~=inf)
end
```

```
%Beginning of path tracking algorithm
for i=1:k %kth shortest path.
     x=1;
     penulti=[];%index vector of successive node's distance.
     scan=[];
 %Starting with the first row of fD matrix(source to sink).
 %Finding eligible nodes in first row of fD satisfying the tracing equation.
    for j=1:size(cho_Adjacent{n,1})
 %Most core part of the path tracing procedure
 penultimate=find(D{1,cho_Adjacent{n}(j)}(1:i)+C(cho_Adjacent{n}(j),n)
                  == D{1,n}(i));

       if ~isempty(penultimate)
          scan(x)=cho_Adjacent{n}(j);
          penulti(x)=penultimate;
          D{1,cho_Adjacent{n}(j)};

          x=x+1;
       end
    end
    %'scan' = list of eligible nodes to be tracked repeatedly in following.


    if ~ismember(1,scan)  %Only the node is not the source node.

        for j=1:size(scan,2)
         [P{i},dis{i}]=pathfinder2(n,penulti(j),cho_Adjacent,D,C,scan(j));
         Path{i}=[P{i},scan(j)];
         distant{i}=[dis{i},C(scan(j),n)];

        end

    else

    Path{i}=1;
    distant{i}=C(1,n);
    end

 Path{i}(size(Path{i},2)+1)=n;
 end
```

```
  tot=[ ];
  for i=1:k
   tot(i)=sum(distant{i});
  end

%Printing results.
for i=1:k
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f %2.0f',Path{i},size(C,2))
 fprintf('\nDistances\n')
 fprintf('%5.2f\t' ,distant{i})
 fprintf('total: %3.2f\n',tot(i))
end
```

A.9 Generalized Minimization Procedure

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Function Matlab code: minimization
 The purpose of this MATLAB program is to perform the geneeralized minimization
 set operation explainde in Chapter 3.4(Double sweep algorithm).

 Associated higher algorithm file
 -chodoublesweep3.m: Double sweep algorithm file.

 Input arguments
   - a,b: Same dimensional vectors

 Output arguments
   - d: Generalizde minimization result vector.

        Author: Capt. Jae Il Cho
        Date  : 11/24/02
 Last Revision: 01/24/03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d=minimization(a,b) c=union(a,b); if size(c,2) <
size(a,2)
    i=size(a,2)-size(c,2);
    c((size(c,2)+1):(size(c,2)+i))=inf;
end d=c(1:size(a,2));
```

A.10 Generalized Addition Procedure

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Function Matlab code: addition
 The purpose of this MATLAB program is to perform the generalized addition
 set operation explainde in Chapter 3.4(Double sweep algorithm).

 Associated higher algorithm file
 -chodoublesweep3.m: Double sweep algorithm file.

 Input arguments
   - a,b: Same dimensional vectors

 Output arguments
   - d: Generalizde minimization result vector.
```

```
        Author: Capt. Jae Il Cho
        Date  : 11/24/02
 Last Revision: 01/24/03
*************************************************************************

function d=addition(a,b) c=[]; k=[]; h=1; for i=1:size(a,2)

  for j=1:size(b,2)
      k(j)=b(j)+a(i);
   end
  c(1,h:(h+size(b,2)-1))=k;
   h=h+size(b,2);
end

c1=unique(c); if size(c1,2) < size(a,2)
    i=size(a,2)-size(c1,2);
    c1((size(c1,2)+1):(size(c1,2)+i))=inf;
end d=c1(1:size(a,2));
```

A.11 Decomposition of Initial Estimate Matrix

```
*************************************************************************
 Function Matlab code: decompose
 The purpose of this MATLAB program is to decompose the initial estimate
 of K shortest path value vector matrix into lower part and upper part.
 However, they are not strictly lower and upper triangular matrix which is
 why we built this code for decomposition (Refer to double sweep algorithm
 in chapter 3.4)

 Associated higher algorithm file
 -chodoublesweep3.m: Double sweep algorithm file.

 Input arguments
   - A: K shortest path estimate cell matrix.

 Output arguments
   - L: Lower triangular style matrix of A.
   - U: Upper triangular style matrix of A.

        Author: Capt. Jae Il Cho
        Date  : 11/24/02
```

```
 Last Revision: 01/24/03
*************************************************************************


function [L,U]=decompose(A)


L=tril(A); U=triu(A);

k=size(A{1,1},2) for i=1:size(A,2)
 for j=1:size(A,2)
   if i==j
    L{i,i}(1:k)=inf;
    U{i,i}(1:k)=inf;
  else if i<j
    L{i,j}(1:k)=inf;
  else
    U{i,j}(1:k)=inf;

   end
 end
end
end
```

A.12 Path Tracing Procedure (Double sweep algorithm)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Function program Matlab code: pathfinder2
 The purpose of this MATLAB program is to find the actual K shortest paths
 with an input of K shortest path distance values vector.  This algorithm is
 to track each of actual K shortest path based on the input of K shortest path
 distance values Chapter 3.4(Double sweep algorithm).

 Associated higher algorithm file
 -chodoublesweep3.m: Double sweep algorithm file.

 Input arguments
  -n = Number of nodes in the graph.
   -k = The index of penultimate node(kth distance of penultimate node).
   -cho_Adjacent = n by 1 cell array in which cho_Adjacent{i} = Adjacent nodes
                   to node i  for all nodes i in the graph.
   -cho_D = Final double sweep cell matrix where cho_D{i,j}= a vector
            having K shortest path distance values from node i to j.
   -cho_C = n by n matrix where cho_C(i,j}= the mean travel time(fixed
            distance) from node i to j.
   -ini = Penultimate node.

 Output arguments
   - p: Node list vector where p(i) is the ith node in path to 'ini'.
   - dis: Distance vector where dis(i) is the distance from i-1th node
          to ith node in the path to 'ini'.

  #This file is explicitly related to higher algorithm
   file,chodoublesweep3.m without which tis file can not work separately.
        Author: Capt. Jae Il Cho
        Date  : 11/24/02
 Last Revision: 01/24/03
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p,dis]=pathfinder(n,k,cho_Adjacent,cho_D,cho_C,ini)

Adjcent=[]; p=[]; x=1; dis=[];

s=ini; Adjacent=cho_Adjacent{ini}

while s~=1

for j=1:size(Adjacent,1)  % For all incident node to penultimate node s.
```

```
 cho_D{1,Adjacent(j)}(1:i);
 penultimate=find(cho_D{1,Adjacent(j)}(1:k)+cho_C(Adjacent(j),s)==cho_D{1,s}(k))
 % penultimate=index of what shortest path in Adjacent(j) node vector.

        if ~isempty(penultimate)
       % if it found the penultimate node and it's not a member which
       % we already found (we need to prevent cycle).
           dis(x)=cho_C(Adjacent(j),s);
           s=Adjacent(j)    %proceeding node
           break
        end

   end

   k=penultimate; % index of what shortest path in penultimate node

   p(x)=s       %Path list
   Adjacent=cho_Adjacent{s}
   x=x+1;
end

%Arrange the order of element of each path and distant vector.
p=p(end:-1:1)

dis=dis(:,end:-1:1);
```

A.13 Asymptotic Stochastic K-shortest Path Heuristic

```
*************************************************************************
 Function program Matlab code: asymptoticpathbig
 This function program code is to compute and find the expected
 K asymptotic, asymptotic expected dependent1,2 shortest paths in a given
 network.  This algorithm differs from 'stochasticpath' in asymptotical
 approaches.  Except this asymptotical approaches, most of measures are
 all the same.

 Algorithm procedure is following.
 Step 0 : Initialization , independent & asymptotic mean travel time of
          each link.
 Step 1 : Double sweep algorithm to find asymptotic K shortest paths.
 Step 2 : Computing the distribution of asymptotic shortest path.
 Step 3 : Computing asymptotically dependent mean travel time of all shortest
          paths.
 Step 4 : Ranking K asymptotically dependent expected shortest paths
           & printing results

 Input arguments
 -A = distance matrix where Aij=if i and j are adjacent,distance between
      node i and j otherwise, Aij=inf
 -D = Traffic condition matrix where Dij= traffic condition of link (i,j).
 -st = Number  of state of the CTMC across the network.
 -kth = K paramter value of K shortest path algorithm, down to kth shortest
        path.

 Output arguements
 -Q = Q matrix cell array where Qij= Q matrix of CTMC of link(i,j)
 -V = Velocity diagonal matrix where Vii= velocity of vehicle
      at state i in CTMC.
 -ASYM = Asymptotic mean travel time matrix where ASYM(i,j)=asymptotic
         mean travel time of link (i,j)
 -INM,ASYDEM,ASYDEM2 = Independent, asymptotic dependent I and
                dependent II mean travel time matrix where INM(i,j),
                ASYDEM(i,j),ASYDEM2(i,j)=mean travel time of each link (i,j).
 -PST = Steady state probability cell matrix where PST{i,j}= steady state
        probability vector  of link (i,j).
 -ADep1dis,ADep2dis,ADep1tot = Distance information storages for K paths of
                          asymptotic dependentI and dependent II.
 -fD = Shortest paths value matrix whose element fD(ij)is a vector of
          k shortest paths from node i to j.
```

```
 -a  = Mean of independent expected shortest path travel time
       distribution.
 -b  = Variance of independent expected shortest path travel time
       distribution
 -aKpath = K shortest path cell array where Kpath{i}= ith shortest path
          nodes storage
 Associated sub algorithm files
   -ex1generation.m file: Generation of Q matrix, steady state probability P
   -vgenerateeq1.m file: Computation of LT{E{T(x)}.
   -invt_lap0.m file: Inversion of LT{E{T(x)}.
   -generateeqk.m file: Computation of asymptotic variance parameter,k.
   -chodoublesweep3.m file: Core algorithm file( K shortest paths)

 Author: Capt. Jae Il Cho
         Date  : 11/24/02
 Last Revision: 01/24/03
*************************************************************************


function [Q,V,ASYM,INM,ASYDEM,ASYDEM2,Z0,PST,aKpath,dis,ADep1dis,
              ADep2dis,ADep1tot,ADep2tot,fD,a,b]=asymptoticpathbig(A,D,st,kth)
n=size(A,2); ASYMDEM=zeros(n,n); ASYMDEM2=zeros(n,n);
ASYMEST=cell(n,n); Q=cell([n,n]); ASYM=zeros(n,n); INM=zeros(n,n);
Z0=cell([n,n]); PST=cell([n,n]);
tic; %Computation time initiated.

%Step 0 : Initialization , independent & asymptotic mean travel time of
%          each link.
%Problem formulation and all initial measures;Mean,Steadystate probability etc.
for i=1:n
  for j=1:n
    if A(i,j)~=inf & i < j

       [Q1,P]=ex1generation(D(i,j),st)
       %[Q1,P]=ex2generation(D(i,j),st)
       z0=zeros(1,st);
       z0(st)=1;
       [LTm1,V]=vgenerateeq1(st,Q1,z0);    %Find the LT{E{T(x)},LT{E[T(x)]^2}.
       m=invt_lap0(A(i,j),LTm1);     %Find the E{T(x)}.
```

```
        %Independent transient mean & variance
        INM(i,j)=m
        INM(j,i)=m


        % Asymptotic mean time.
        ASYM(i,j)=A(i,j)/sum(P*V);
        ASYM(j,i)=ASYM(i,j);

        %steady-state distribution & Q matrix.
        PST{i,j}=P;
        PST{j,i}=P;
        Q{i,j}=Q1
        Q{j,i}=Q1;


        end


     end
   end
end




%Step1: Asymptotic k shortest path.
 for i=1:n
    for j=1:n
     if INM(i,j)==0
        INM(i,j)=inf;
    end
 end
end
%Computing the expected state vector of initial arcs.
for j=1:n
 if ASYM(1,j)~=0
   ASYMEST{1,j}=zeros(1,st);
   est=round(PST{1,j}*[1:st]');
   ASYMEST{1,j}(1,est)=1;
end end [L,U,fD,aKpath,dis,tot]=chodoublesweep3(ASYM,kth);
```

```
%Step2: Asymptotic mean and variance.
adistrimean=tot(1); adistrivari=0;
for i=1:size(aKpath{1},2)-1
[ltk]=
generateeqk(V,Q{aKpath{1}(i),aKpath{1}(i+1)},PST{aKpath{1}(i),aKpath{1}(i+1)});
ASYMVARI(aKpath{1}(i),aKpath{1}(i+1))=A(aKpath{1}(i),aKpath{1}(i+1))*ltk;
adistrivari=adistrivari+ASYMVARI(aKpath{1}(i),aKpath{1}(i+1));
end


%Step3: Asymptotic dependent travel time.

%Computing of expected dependent travel time of each selected shortest path from
% K dominant paths.
ADep1dis=cell([1,kth]); ADep2dis=cell([1,kth]);
ADep1tot=zeros(1,kth); ADep2tot=zeros(1,kth); for i=1:kth
    %Starting link mean travel time.
    ADep1dis{i}(1)=INM(aKpath{i}(1),aKpath{i}(2))
    ADep2dis{i}(1)=INM(aKpath{i}(1),aKpath{i}(2))
    for j=2:size(aKpath{i},2)-1

     g=aKpath{i}(j);
     h=aKpath{i}(j+1);

    %Asymptotic Expected probability at E[T(x)]:dependent I
    z0=PST{aKpath{i}(j-1),aKpath{i}(j)}; %PST: steady state probabilitiy vector
    [LTe]=vgenerateeq(st,Q{g,h},z0);
    ASYMDEM(g,h)=invt_lap0(A(g,h),LTe); %Dependent I mean travel time of link (g,h)

    %Asymptotic Expected state at E[T(x)]:dependent II
    z02=ASYMEST{aKpath{i}(j-1),aKpath{i}(j)};
    %ASYMEST : Asymptotic Expected state approach probability vector
    [LTe]=vgenerateeq(st,Q{g,h},z02);
    ASYMDEM2(g,h)=invt_lap0(A(g,h),LTe);

    ASYMEST{g,h}=zeros(1,st);
    est=round(PST{g,h}*[1:st]');
    ASYMEST{g,h}(1,est)=1;
```

```
    ADep1dis{i}(j)=ASYMDEM(g,h);
    ADep2dis{i}(j)=ASYMDEM2(g,h);

   end
   ADep1tot(i)=sum(ADep1dis{i});
   ADep2tot(i)=sum(ADep2dis{i});

 end


 %Step4: Ranking procedures.

  %Sort and keep indices of rank.
  [sorted1,index1]=sort(ADep1tot);
  [sorted2,index2]=sort(ADep2tot);



 t=toc; %Computation time terminated.
 %We are done and happy to print out all outputs needed!
 fprintf('Asymptotic mean  Case!')
 for i=1:kth
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',aKpath{i})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,dis{i})
 fprintf('Asymptotic total travel time :total: %3.4f\n',tot(i))
end

 disp('')
 disp('')
fprintf('Asymptotic Dependent I Case!')
 for i=1:size(index1,2)
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',aKpath{index1(i)})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,ADep1dis{index1(i)})
 fprintf('Asymptotic Dependent I total travel time:
          %3.4f\n',ADep1tot(index1(i)))
 end
 disp('')
 disp('')
```

```
fprintf('Asymptotic Dependent II Case!') for i=1:size(index2,2)
 fprintf('\n%1.0f st path nodes\n',i)
 fprintf('%2.0f ',aKpath{index2(i)})
 fprintf('\nDistances\n')
 fprintf('%5.4f\t' ,ADep2dis{index2(i)})
 fprintf('Dependent II total travel time : %3.4f\n',ADep2tot(index2(i)))
 end

fprintf('The asymptotic expected shortest path distribution\n')
adistrimean adistrivari a=adistrimean; b=adistrivari;
fprintf('computational time') t
```

A.14 Asymptotic Variance Parameter

```
%************************************************************************
 Function Matlab code: generateeqk
 The purpose of this MATLAB code is to compute the asymptotic variance
 paramter k which is used to compute the asymptotic variance(chapter 2).

 Associated higher algorithm file
  - asymptoticpathbig.m

 Input arguments
   - V: Diagonal velocity matrix.
   - Q: Q matrix.
   - P: Steady state probability distribution.

 Output arguments
   - k: Asymptotic variance parameter,k.

        Author: Capt. Jae Il Cho
        Date  : 11/24/02
 Last Revision: 01/24/03
%************************************************************************

function [K]=generateeqk(V,Q,P)

syms s
av=sum(P*V); % P*V =expected asymptotic velocity
D=inv(V)*Q; iV=inv(V);
[Rv,Rei]=eig(D) ;% Right eigen vectors,values
[Lv,Lei]=eig(D.'); %Left eigen vectores,values

n=size(D,1);

%Making new left eigen vectors which exactly matches to same right eigen values.
Lvn=zeros(n,n);



Drei=diag(Rei);%Diagonal element
Lrei=diag(Lei);

for i=1:n
%To maintain allowable accuracy while matching the left eigenvector to the same
```

```
%location of right eigenvector with same eigenvalue,
%we had to make difference vector,Dei.
   if Rei(i,i) ~= 0

      Dei=Lrei-Rei(i,i) ;
   [h]=find(-0.00001<Dei & Dei<0.00001);%Acceptable accuracy
   %Making new lefteigen vector exactly corrsponding to same right eigen value.
   Lvn(:,i)=Lv(:,h);
  end
end
Lvnew=Lvn';  % Transpose. For computational purpose,product sum computation.


%Finding K value as followings.
su=[];
e1=ones(n,1); % e vector(summation)
for i=2:n
 su(i)=(1/Rei(i,i))*(((P*Rv(:,i))*(Lvnew(i,:)*iV*e1))/(sum(Lvnew(i,:)*Rv(:,i))));
end

K=-(2/av)*(sum(su));
```

Appendix B. Cycle Path Reduction in Double Sweep Algorithm

While we implement the double sweep algorithm, intrinsically we can encounter unwanted cycle shortest path. Particularly for the network that the triangular inequality does not hold as follows, For any directly connected three nodes in a graph

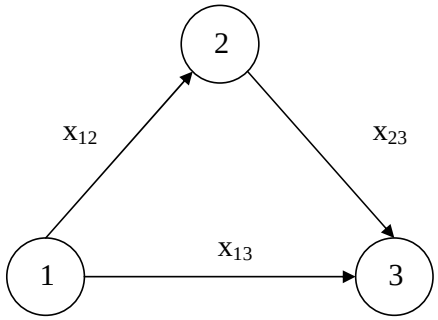

Figure B.1 Triangular inequality

$G(N, E)$ $\{i, j, k\}$, the triangular inequality is defined as

$$(i, j) < (i, k) + (j, k)$$

In the graph (Figure B.1), if $x_{13} < x_{12} + x_{23}$ then the triangular inequality holds.

Thus, for a graph that does not satisfy the triangular inequality for all links the cycle path could be the shorter than non-cycle path. Even though the cycle path is actually shorter than non-cycle path we may not want to have it as the solution. Hence, in order to prevent this intrinsic cycle path from happening in double sweep algorithm we made slight modification. While iteratively updating the distance estimate vector d_{ij}^r at r th iteration, we purposedly do not compute the self distance estimate for all node. In other words, we do not update the estimate of d_{ii}^r for all node i.

Hence while we do backward and forward sweep, we skip all diagonal elements of estimate array matrix $[d_{ij}]$ where d_{ij} is the K-vector of shortest path estimates.

However, we still may have shorter cycle path in those graph where the triangular inequality does not hold. But at least we reduce the computational time and cycle path occasions.

Appendix C. Dependence on Distinct CTMC Sample Space

While each link's CTMC sample space might be distinct, the velocity function across the network is assumed to be same. Thus, the basic idea of imposing the dependence on terminal condition of a distinct CTMC sample space lies on how to maintain legitimate velocity transition from the preceding condition to the following condition.

First, let us consider the case where a vehicle enters into a smaller number of CTMC sample space from the larger number of CTMC sample space. If the expected terminal state of this vehicle in the preceding is beyond the sample space of the following link, then under totally new environment (CTMC) this vehicle can not exceed the maximum state of the following CTMC's space. For numerical example, if $E[Z_{ij}(T(x_{ij}))] = 5$ where $S_{ij} = \{1, 2, .., 5\}$ and $S_{jk} = \{1, 2, 3\}$, the initial condition under which a vehicle's velocity is governed is $P\{Z_{jk}(0) = 3\} = 1$. The initial state of CTMC cannot beyond the sample of its CTMC ($Z_{jk}(0) \not> 3$). In terms of velocity, suppose $V = 100/Z(T)$, the terminal velocity was 20 and the initial velocity in new environment will be 33 because it is the maximum speed that a vehicle can have in a new environment. This matching concept leads us to the following methodologies of imposing the dependence across distinct sample spaces (Figure C.1 and Figure C.2).

Suppose there are two successive link $(i, j), (j, k)$ and distinct environment process CTMC of each link where sample space are distinct, $S_{(i,j)} \neq S_{(j,k)}$. Then, we impose dependence of expected terminal distribution and expected terminal state on subsequent link's CTMC initial distribution ($[Z_{jk}(0)]$) as the following Figure C.1 and Figure C.2. For each of methodologies, we consider two cases that the terminal sample space is larger and initial sample space is larger.

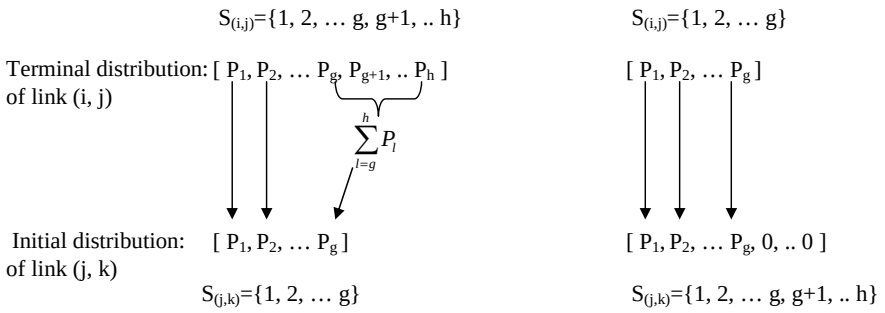

Figure C.1 Expected terminal distribution approach

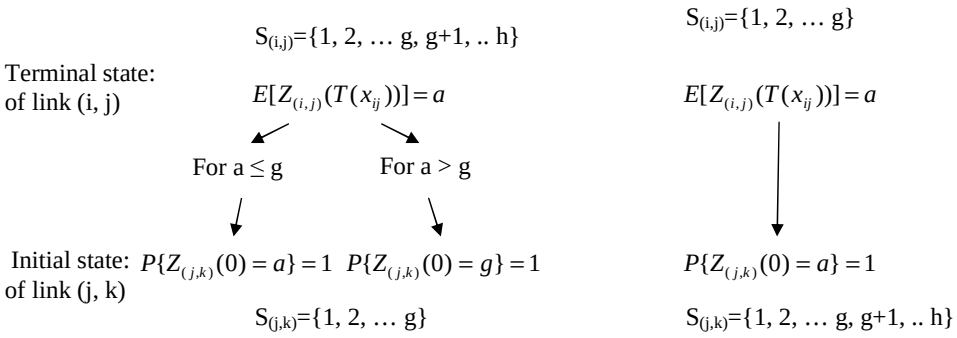

Figure C.2 Expected terminal state approach

Since the sample space of environment process CTMC reflects exactly the possible velocity range of its terrain (link), the above matching procedure capture the reality of transition to distinct environment while it seems to occur discontinuously.

Appendix D. Q Matrices

D.1 Q matrices for the Expected Shortest Path Problem I

$Q\{i,j\} := Q$ matrix of CTMC of link (i,j) and $Q\{i,j\} = Q\{j,i\}$.

```
Q{2,1} =

 -253.6056  247.9687    5.6369
  261.5909 -268.0437    6.4528
  268.5140  251.0153 -519.5293



Q{3,1} =

 -236.8661  231.7751    5.0910
  213.0202 -222.1647    9.1445
  266.7846  201.3626 -468.1472



Q{4,1} =

 -173.0152   73.8296   99.1856
   96.1175 -178.7337   82.6162
   88.6340   55.3088 -143.9428



Q{3,2} =

  -49.1639   20.1372   29.0268
    0.0907   -0.1411    0.0504
    0.0759    0.0331   -0.1090



Q{5,2} =
```

```
  -12.6397    7.7987    4.8410
    0.0943   -0.1680    0.0737
    5.7961    6.6650  -12.4612
```

```
Q{4,3} =

  -69.8975   39.8190   30.0786
    0.0219   -0.1007    0.0788
    0.0105    0.0457   -0.0562
```

```
Q{5,3} =

 -294.5767  290.8875    3.6892
  258.8739 -265.9259    7.0519
  231.3435  223.1159 -454.4594
```

```
Q{5,4} =

   -6.8185    5.6296    1.1889
    0.0479   -0.1271    0.0793
    4.8559    9.5222  -14.3781
```

D.2 Q matrices for the Expected Shortest Path Problem II

$Q\{i,j\} := Q$ matrix of CTMC of link (i,j) and $Q\{i,j\} = Q\{j,i\}$.

```
Q{2,1} =

  -41.8991   11.5569   30.3421
   24.2991  -62.4040   38.1048
   22.8234    0.9252  -23.7486
```

```
Q{3,1} =

 -353.3695   80.7716   89.5969   96.0906   86.9104
   58.8133 -321.9455   96.7735   95.8452   70.5135
   94.6825   52.8946 -288.7284   90.6583   50.4931
   56.9445   60.1383   59.9361 -240.6283   63.6094
   59.9407   50.7637   87.3393   72.2548 -270.2985
```

```
Q{4,1} =

  1.0e+003 *

   -0.9503    0.1419    0.1846    0.1525    0.1203    0.1672    0.1838
    0.1020   -0.8872    0.1379    0.1832    0.1503    0.1709    0.1429
    0.1305    0.1190   -0.8172    0.1682    0.1303    0.1542    0.1151
    0.1698    0.1378    0.1860   -0.9926    0.1594    0.1497    0.1900
    0.1822    0.1645    0.1818    0.1660   -0.9576    0.1290    0.1341
    0.1534    0.1727    0.1309    0.1838    0.1568   -0.9680    0.1703
    0.1547    0.1445    0.1695    0.1621    0.1795    0.1957   -1.0059
```

```
Q{3,2} =

 -283.8240   58.6478   98.9873   63.5724   62.6165
```

```
   93.7871 -295.8958   56.8259   50.5878   94.6949
   59.9569   64.9362 -262.5747   64.2204   73.4612
   53.2391   99.4167   79.1396 -307.5710   75.7756
   66.6976   71.6453   61.2975   78.9903 -278.6307
```

```
Q{6,2} =

  -42.4798   32.0263   10.4535
   18.9909  -53.0332   34.0423
   23.0548   28.3914  -51.4462
```

```
Q{6,3} =

 -826.2888  160.2869  105.0269  141.5375  130.4999  187.4367  101.5009
  176.7950 -969.7828  199.0083  178.8862  143.8659  149.8311  121.3963
  164.3492  132.0036 -911.4626  172.6632  141.1953  174.4566  126.7947
  143.9924  193.3380  168.3332 -965.8432  183.9238  162.8785  113.3773
  120.7133  160.7199  162.9888  137.0477 -831.0017  145.1425  104.3895
  102.7185  131.2685  101.2863  138.3967  168.3116 -745.5154  103.5338
  161.2395  160.8540  101.5760  101.6355  119.0075  158.6918 -803.0044
```

```
Q{7,3} =

 -306.2917   81.5726   85.8817   84.6335   54.2040
   72.7178 -281.8428   67.6625   57.6803   83.7822
   84.9607   86.3755 -305.1306   77.7421   56.0524
   72.5377   85.7941   94.6421 -315.7124   62.7385
   93.2802   61.6175   90.2436   95.4199 -340.5612
```

```
Q{5,4} =
```

```
   -6.4069    2.4877    3.9192
   32.0408  -74.2342   42.1935
    8.6950    8.5396  -17.2347
```

```
Q{7,5} =

 -812.3855  134.0048  131.4217  136.5078  139.3240  159.1525  111.9747
  103.8129 -913.9837  186.9867  193.4237  126.4449  116.0300  187.2855
  123.7880  164.5831 -856.5961  166.4931  187.0381  100.9927  113.7010
  181.8756  143.0166  189.0322 -933.8714  168.7324  134.6112  116.6035
  115.5613  119.1116  142.2452  185.5976 -890.1861  181.5935  146.0770
  145.7354  145.0689  141.2219  190.1610  100.5584 -827.6619  104.9162
  169.3180  165.0106  198.2988  155.2673  140.0074  119.8789 -947.7810
```

```
Q{7,6} =

 -277.9645   68.7943   50.4938   70.9929   87.6835
   89.6936 -331.3574   92.2361   68.3876   81.0401
   86.5639   59.6947 -306.3083   78.4603   81.5895
   61.7206   77.4391   96.5792 -318.5155   82.7766
   69.5952   81.3657   84.9540   69.8592 -305.7742
```

Bibliography

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

2. D.R.Shier (1978). Iterative methods for determining the k shortest paths in a network. *Networks*, **6**, 205-225.

3. D.R.Shier (1979). On algorithms for finding the k shortest paths in a network. *Networks*, **9**, 195-214.

4. C. Elliott Sigal, A. Alan B. Pritsker and James J. Solberg (1979). The stochastic shortest path problem. *Operations Research*, **28**, 1122-1128.

5. Jerzy Kamburowski (1985). A note on the stochastic shortest path route problem. *Operations Research*, **33**, 696-698.

6. Amir Eiger, pitu B. Mirchanani and Hossein Soroush (1985). Path preferences and optimal paths in probabilistic networks. *Transportation Science*,**1**, 75-84.

7. Jane N. Hagstrom (1990). Computing the probability distribution of project duration in a PERT network. *Networks*, **20**, 231-244.

8. Dimitri P. Bertsekas and John N. Tsitsilkis (1991). An analysis of stochastic shortest path problems. *Mathematics of Operations Research*,**16**, 580-595.

9. James R. Evans and Edward Minieka (1992). *Optimization Algorithms For Networks and Graphs.* 2nd ed., Marcel Dekker,Inc., New York.

10. Gehen A. Corea and Vidyadhar G. Kulkarni (1993). Shortest paths in stochastic networks with arc lengths having discrete distributions. *Networks*, **23**, 175-183.

11. Harilaos N. Psafatis and John N. Tsitsiklis (1993). Dynamic shortest paths in acyclic networks with Markovian arc costs. *Operations Research*, **41**, 91-101.

12. Ravindra K. Ahuja, Thomas L. Magnanti and James B. Orlin (1993). *Network Flows Theory, Algorithms, and Applications.* Prentice Hall, New Jersey.

13. JenYeng Cheah and J. MacGregor Smith (1994). Generalized M/G/C/C state dependent queueing models and pedestrian traffic flows. *Queueing Systems*,**15**, 365-386.

14. Warren B. Powell and Raymond K. Cheung (1994). Stochastic programs over trees with random arc capacities. *Networks*, **24**, 161-175.

15. Vidyadhar G. Kulkarni (1995). *Modeling and Anaysis of Stochastic Systems.* Chapman & Hall, New York.

16. M.O. Ball, T.L. Magnati, C.L. Monma and G.L Nemhauser,eds (1995).Stochastic and dynamic networks and routing. *Handbook in Operations Research and Management Science. Networks*,**4**,141-295.

17. Ishwar Murthy and Summit Sarkar (1996). A relaxation-based pruning technique for a class of stochastic shortest path problmes. *Transportation Science*, **30**, 220-236.

18. Raymond K. Cheung and Warren B. Powell (1996). An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management. *Operations Research*, **6**, 951-963.

19. Rajat Jain and Macgregor Smith (1997). Modeling vehicular traffic flow using M/G/C/C state dependent queueing models. *Transportation Science*, **31**, 324-336.

20. Liping Fu and L.R. Rillet (1998). Expected shortest paths in dyanmic and stochastic traffic networks. *Transportation Research*, **32**, 499-516.

21. Elise D. Miller-Hooks and Hani S. Mahmassani (1998). Least possible time paths in stochastic time-varying networks. *Computers and Operations Research*, **25**, 1107-1125.

22. Ishwar Murthy and Summit Sarkar (1998). Stochastic shortest path problems with piecewise-linear concave utility functions. *Management Science*,**44**, s125-s136.

23. Elise D. Miller-Hooks and Hani S. Mahmassani (2000). Least expectd time paths in a stochastic, time-varying trasnportation network. *Transportation Science*, **34**, 198-215.

24. Jeffrey P. Kharoufeh and Natarajan Gautam (2002). First passage time analysis of a moving particle in a random environment. Working Paper, Dept. of Operational Science, Air Force Institute of Technology.

25. Jeffrey P. Kharoufeh and Natarajan Gautam (2002). Deriving link travel time distributions via stochastic speed processes. To appear in *Transportation Science*.

26. Amir Azaron and Farhad Kianhar (2002). Dynamic shortest path in stochastic dynamic networks: Ship routing problem. *European Journal of Operational Research*, **144**, 138-156.

Vita

Captain Jae Il Cho entered Republic of Korea Military Academy in 1991. He graduated from the Academy in March of 1995 with a Bachelor of liberal arts in French. Upon graduation, he was commissioned as a 2^{nd} Lieutenant, Army Armor Officer. After a six-month Basic Officer Course in Army Mechanized Unit School in Kwang-ju Korea, he was assigned to 3^{rd} Armored Brigade as an tank platoon leader. After serving one year, he was selected for an assignment in Capital Defense Command, Seoul and was assigned in Presidential Guard regimental unit as an armored platoon leader. His platoon was a strike force for Presidential Guard. He then attended the Advanced Officer Course for company commander in Army Mechanized Unit School.

He served as a tank company commander in 3^{rd} Infantry Division deployed along DMZ. He commanded the most front line Tank company for 18 months. Upon completion of company commander, he applied for annual Army Graduate Education Plan and entered the Air force Institute of Technology for Systems Engineering program, USA in August 2001. He was inducted into Tau Beta Pi in his 4^{th} quarter in AFIT. Upon graduation, he will be assigned to field division in Korea.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
25-03-2003	**Master's Thesis**	Apr 2002 – Mar 2003

4. TITLE AND SUBTITLE

SHORTEST PATH PROBLEMS IN A STOCHASTIC AND DYNAMIC ENVIRONMENT.

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Cho, Jae, I., Captain, Republic of Korea Army

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GSE/ENS/03-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this research, we consider stochastic and dynamic transportation network problems. Particularly, we develop a variety of algorithms to solve the expected shortest path problem in addition to techniques for computing the total travel time distribution along a path in the network. First, we develop an algorithm for solving an independent expected shortest path problem. Next, we incorporate the inherent dependencies along successive links in two distinct ways to find the expected shortest path. Since the dependent expected shortest path problem cannot be solved with traditional deterministic approaches, we develop a heuristic based on the K-shortest path algorithm for this dependent stochastic network problem. Additionally, transient and asymptotic versions of the problem are considered. An algorithm to compute a parametric total travel time distribution for the shortest path is presented along with stochastically shortest path measures. The work extends the current literature on such problems by considering interactions on adjacent links.

15. SUBJECT TERMS

Stochastic networks, Shortest path problem, K shortest paths.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jeffrey P. Kharoufeh (ENS)
U	U	U	UU	158	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4336; e-mail: Jeffrey. Kharoufeh@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18