Routing Problems in Stochastic Time-Dependent Networks with Applications in Dynamic Traffic Assignment

by

Song Gao

B.S., Civil Engineering
Tsinghua University (1999)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2002

Author ....
Department of Civil and Environmental Engineering
January 18, 2002

Certified by...
Ismail Chabini
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by ....
Oral Buyukozturk
Chairman, Departmental Committee on Graduate Studies

Routing Problems in Stochastic Time-Dependent Networks with Applications in Dynamic Traffic Assignment

by

Song Gao

Submitted to the Department of Civil and Environmental Engineering
on January 18, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

Stochasticity is prevalent in transportation networks in general, and traffic networks in particular. The overall objective of this thesis is to study implications and significance of stochasticity in the development of models and algorithms for dynamic traffic flows in road networks. There are two major parts in this thesis. We first study the best routing policy problems in stochastic and time-dependent networks, and then develop policy-based stochastic dynamic traffic assignment models and algorithms.

Routing problems are not only useful to develop dynamic traffic assignment (DTA) methods, but are also fundamental network optimization problems with a wider application domain. We define the problem in general and give a framework, which we believe is the first in the literature. We give a comprehensive taxonomy and an in-depth discussion of most of the variants of the problem. We study in detail a variant pertinent to the traffic in road networks. We give an exact solution algorithm to this variant, analyze its running time complexity and point out the importance of finding good approximation algorithms. We then present several approximations, and study their effectiveness against the exact algorithm, both theoretically and computationally.

We proceed to develop a policy-based stochastic dynamic traffic assignment model. We give a conceptual framework and then develop models for users' choice of policies and the dynamic network loading problem. These models are two major components of the overal DTA model. We give solution algorithms for these models, and present a heuristic algorithm to solve the proposed policy-based DTA model. Using an example, we show that policy-based DTA models have solutions that are different in expected travel times than the path-based models.

Thesis Supervisor: Ismail Chabini
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgments

I would like to thank Professor Ismail Chabini for his guidance and support during my stay at MIT. He never loses confidence in me even when myself does. He has endless energy and ideas, making the exploration of research problems full of excitement. I enjoy very much the many discussions we have had about research and about life. Thanks also to Professor Moshe E. Ben-Akiva for his research supervision in my first year and Professor Cynthia Barnhart for her advice on my academic development.

Thanks to National Science Foundation for providing me the freedom to study fundamental research problems and the opportunity to present my work to a large research community. Special thanks to Jon Bottom for introducing me to the domain of fixed-point problem and the series of Sim^2 tutorial. Thanks to the ACTS group for listening to my horrible presentations and their helpful advice on my research. Thanks also to my dear friends of ACCA: Wei, Jingsong, Delin, Yi, Haohao, Jianyi, Jingjing, Xingxing, Minqu, and ACCA's honorable friend Peilei for making my life at MIT colorful.

Finally I would like to thank my parents for always being there. This work is dedicated to them.

To my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Road traffic congestion is a significant problem of modern society. The effects of congestion impact work trips, as personal trips and freight trips alike. The impacts of congestion are multi-fold. To an individual traveler, congestion reduces the quality of life by consuming one's leisure time, increasing anxiety, and wasting personal resources. To firms, congestion reduces the work efficiency of employees and increases freight transportation costs. To the society as a whole, congestion deteriorates environmental quality by causing more gas emissions and noise, and endangers traffic safety by causing drivers' psychological disturbances.

It is commonly recognized that building more infrastructure, which is usually politically, financially, and environmentally constrained, is not the only remedy to congestion. Furthermore, new infrastructure will induce more demand, which could affect the increased capacity or even make the congestion worse. In nowadays, traffic management and control measures to relieve congestions are generally based on the concept of making maximum use of current infrastructure. These measures can be categorized as from either demand side or supply side. Those from demand side include taking alternative traffic modes and taking alternative ways of work, such as e-conference. Those from the supply side aim at improving traffic flows by making use of advances in information technology, which is the underlying idea of the Intelligent

Transportation Systems (ITS).

ITS requires Traffic models that are different from the traditional planning models. In such models, traffic variables are represented in a much smaller time scale than those used for traditional planning purpose. For example, an analysis period of 1.5 hours during the peak hours is usually discretized into 90 time steps with 1 minute per step, and traffic flows can be modeled as functions of time step. This time-dependency of traffic models is required, because real-time traffic measurements are available and real-time decisions are desired. A number of dynamic traffic models have been built, with corresponding algorithms and implementations. Among them the dynamic traffic assignment model is a very important one. The dynamic traffic assignment model takes as input the time-dependent O-D trips and a given traffic network, and outputs time-dependent traffic variables, such as link travel times, O-D travel times and link volumes, which are needed to make real-time control and/or management decisions. Dynamic traffic assignment (DTA) models are the intelligent core of ITS applications and are the topic of many ongoing research projects.

Another feature of traffic models required by ITS applications, yet less recognized and less studied by the transportation research community, is the ability to model stochasticity. In traffic applications, many variables are typically known *a priori* at best with uncertainty. The uncertainty is due to multiple sources. One source of uncertainty is the imperfect data and limited modeling abilities. Data on O-D trips, for example, is hardly known perfectly and deterministically. We might have an average pattern for O-D trips from historical data, but the possibility of deviation from the average pattern always exists. The magnitude of deviation cannot be known deterministically *a priori*, and can only be revealed as informative traffic measurements are available. Data on traffic network supply is also usually uncertain. Bad weather is a major cause of traffic congestion, while weather forecast are usually made at best with errors. Accordingly, the reduction in capacity induced by bad weather is also not deterministic. Unlike bad weather which is predictable to some extent, traffic accidents and incidents are typically unpredictable, making the stochasticity in network supply even more significant. Besides imperfect data, the limited modeling ability is another important source of stochasticity. An example is the modeling of drivers' behavior. Classical behavioral models are naturally stochastic, as all factors affecting a driver's decisions cannot be captured in the model. Ben-Akiva [3] gives four causes of random errors in drivers' behavioral models.

Another source of uncertainty is the random factors in implementing models. Bottom [6] does a comprehensive survey on this issue. Several sources of stochasticity from the model implementations are studied, including rounding data to integers and randomizing the order in processing of network elements. Gibbs sampling, a method to obtain samples from a stochastic process with full conditional probabilities, is used to study link volumes for the test network.

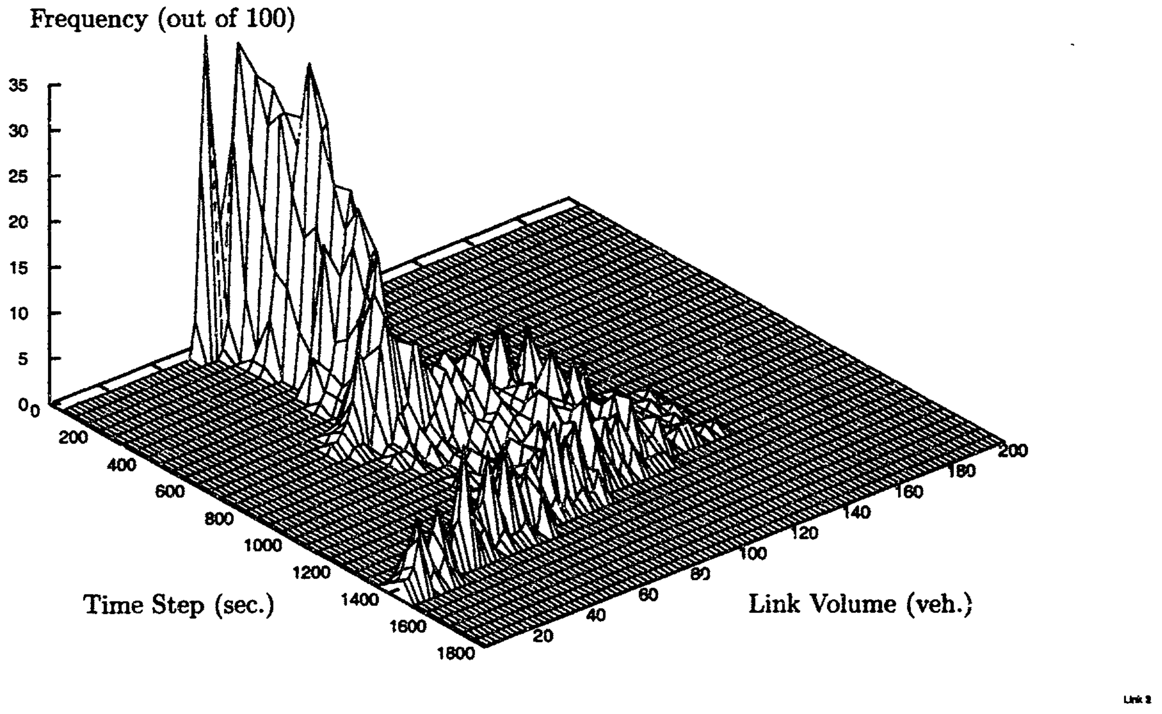


Figure 1-1: Link Volume of Link 2 by Gibbs Sampling (borrowed from Bottom [6] p.162)

Figure 1-1 shows a profile of link volume for one of the links. We can observe a certain level of stochasticity in the link volume distribution. During a certain period (e.g. time step 800-1200), the distributions spread over a relatively large range and it may not be valid to assume the volume is a deterministic value perturbed by noises, which is a prevailing assumption in most of the dynamic traffic models in the current literature. This can be viewed as a proof of stochasticity due to model implementations.

Unlike the time-dependency of traffic models which has been fully modeled in nearly all current ITS applications, stochasticity is recognized but rarely explicitly modeled. In the literature, it is realized that there exist random factors in many aspects of a system, but usually a deterministic approximation of the random system is adopted, due to the difficulty in explicitly modeling stochasticity.

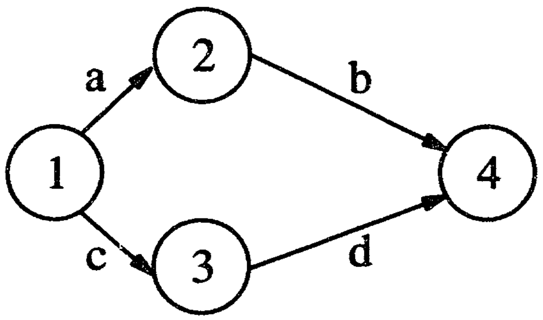
A DTA model is usually composed of three individual models: the users' behavioral model, the dynamic network loading model, and the routing model. The three components interact with each other. As discussed before, users' behavioral models generally output choice probabilities for available alternatives such as paths. However, when these probabilities are taken as input to a dynamic network loading model, they are transformed into population share fractions for paths, which is only a large-sample approximation of the choice probabilities. A dynamic network loading model is to load the trips to their selected paths in a given network and output the resulting link travel times and other traffic variables of interest. We have already seen that the network supply could be very stochastic, yet current network loading models do not consider the stochasticity: in an analytical network loading model, the equations that define the loading processes are deterministic; and in a simulation network loading model, generally the average of link travel times over individual vehicles and over different simulation runs is taken as the output, rather than the random profiles of the link travel times. These deterministic link travel times are then taken as input to the routing model where deterministic dynamic shortest path algorithms are applied.

In this thesis, traffic variables are explicitly modeled as time-dependent random variables. Specifically, we define a stochastic time-dependent (STD) network as a network whose link travel times are random variables with time-dependent distributions. Traffic models that work with random traffic variables have the potential to be more realistic than their deterministic counterparts, as they capture more characteristics of the traffic system. However, whether and to which extent stochastic traffic models are superior to deterministic models is not known yet. This question cannot be answered until stochastic traffic models are built and tested against real-world data. One such traffic model of interest is the stochastic dynamic traffic assignment model. Due to the central role of a dynamic traffic assignment model to ITS applications, the study of the implications and significance of stochasticity for a DTA model is of great importance.

In this thesis, we first study in depth the routing model in a given stochastic dynamic traffic network. We use routing to denote the action to move entities from one location to another in the given network. This is usually done such that a given criterion is optimized, such as travel cost or travel reliability. Routing in a stochastic and time-dependent network is also a problem of fundamental research significance and has a wide domain of applications. In this thesis, the best routing policy problems in stochastic and time-dependent networks, abbreviated as "the BRP problems in STD networks", are studied. A routing policy is a decision rule that specifies which node to take next at each decision node based on current time and realized network link travel times. A best routing policy (BRP) is a routing policy that moves a traveler on a network from one node to another in least expected travel time. Mathematical definitions of these terms can be found in Chapter 2 in this thesis. In fact there might be various criteria to determine a best routing policy. For instance, in addition to expected travel times, the variability of travel times can also be an important factor in a traveler's routing decision. Consequently depending on the traveler's attitude to risk, variability may play different role in the routing decision-making. After building the routing model, we then proceed to the other two components: the users' behavioral model and the dynamic network loading model. Both of the two models are based on routing policies. The three components are then integrated into a stochastic dynamic traffic assignment model.

1.2 Path vs. Routing Policy

The focus of our discussion has been on routing policy. In fact, routing in a STD network can take one of the two forms: a path and a routing policy. A path is a pre-specified set of successive links between a pair of nodes. Travelers who follow a path make decisions *a priori* and take a fixed set of links, regardless of the network conditions. In contrast, travelers who follow a routing policy make decisions en route and therefore can end up taking different set of links, depending on the network conditions that have been revealed during their trip up to the current time. Two illustrative examples in Figure 1-2 and Figure 1-3 will show the difference between a path and a routing policy.



$$(t_a, t_b, t_c, t_d) = \begin{cases} (5, M, 1, 9) & w.p.\ 0.5 \\ (1, 6, 4, M) & w.p.\ 0.5 \end{cases}, \text{ where } M \text{ is a very large positive number}$$

Figure 1-2: Paths vs. Routing Policies in a Non-Time-Dependent and Statistically Dependent Network

In Figure 1-2, we seek to do routing from node 1 to node 4. Two paths, *a*-*b* and *c*-*d*, are available from node 1 to node 4. The network is stochastic but not time-dependent. The data shown beneath the network shows the distribution of travel times of different parts of the network. From this data, we can see that one of the paths will be blocked at a given time. For instance, link *b* can have a very large travel time M with probability 0.5. Assume that one can learn the actual realization of travel times of link *a* and link *c* when one arrives at node 1.

We can compute the expected travel times of path $a - b$ and path $c - d$. The expected travel time of path a-b is $(5+M) \times 0.5 + (1+6) \times 0.5 = 6 + M/2$, and that of path c-d is $(1+9) \times 0.5 + (4+M) \times 0.5 = 7 + M/2$. If the routing model from node 1 to node 4 outputs least expected travel time paths, one will always choose path a-b. However, one can do better if the information we collected at node 1 is adequately explored. If the routing model outputs least expected travel time routing policies, we will do the following: when the travel time of link a is 2, we choose path c-d; otherwise we choose path a-b. The expected O-D travel time of the policy is $(1+9) \times 0.5 + (1+6) \times 0.5 = 8.5$. A best routing policy defers the decision until some useful information is collected. In this example, the decision is delayed until one knows which path is blocked.

This example shows the usefulness of information in a stochastic routing problem. The value of information in the example is due to the statistically dependency of link travel times. When we learn the realizations of some of the links, we can make inferences about travel times of other links so that better decisions can be made. There are cases where the link travel times are not statistically dependent, but their time-dependency makes information valuable. An example in Figure 1-3 shows the case.

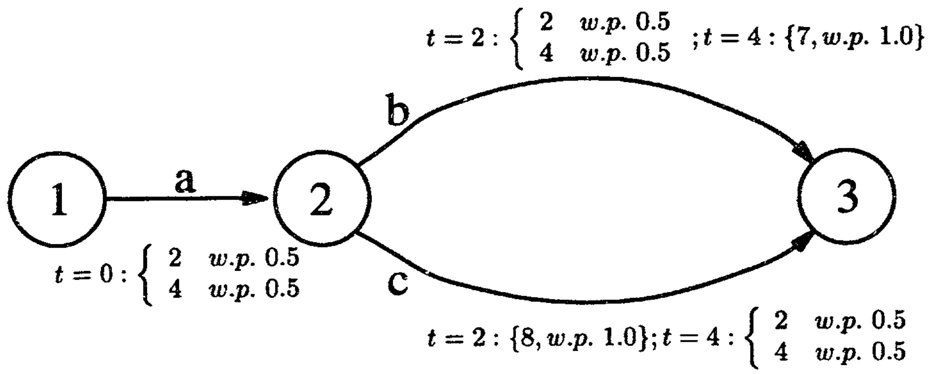

$t = 2 : \begin{cases} 2 & w.p.\ 0.5 \\ 4 & w.p.\ 0.5 \end{cases}; t = 4 : \{7, w.p.\ 1.0\}$
b
1
a
2
3
$t = 0 : \begin{cases} 2 & w.p.\ 0.5 \\ 4 & w.p.\ 0.5 \end{cases}$
c
$t = 2 : \{8, w.p.\ 1.0\}; t = 4 : \begin{cases} 2 & w.p.\ 0.5 \\ 4 & w.p.\ 0.5 \end{cases}$

Figure 1-3: Paths vs. Routing Policies in a Time-Dependent and Statistically Independent Network

Data next to each link shows the probability mass function (PMF) of the link travel time for that link at each departure time of interest. Let us assume in this example that the link travel time random variables are statistically independent. We also assume that only the arrival times are available to the traveler. This implies, among others, that the traveler does not know the actual realizations of link b or link c at node 2. This assumption is made in order to show that the knowledge of arrival times can also benefit the routing decision-making in a STD network. We seek to do routing from node 1 to node 3 for departure time 0. The least expected travel time path is path a-b, with an expected travel time of $(2+2)\times 0.25+(2+4)\times 0.25+(4+7)\times 0.5 = 8$, while the expected travel time of path a-c is $(2+8)\times 0.5+(4+2)\times 0.25+(4+4)\times 0.25 = 8.5$.

Let us consider the following routing policy: when the arrival time at node 2 is 2, take link b as next link; if the arrival time at node 2 is 4, take link c as next link. The expected travel time of the routing policy is $(2+2)\times 0.25+(2+4)\times 0.25+(4+2)\times 0.25+(4+4)\times 0.25 = 6$. The decision in this routing policy is delayed until the arrival time at a decision node is known.

An intuitive representation of the routing policy, denoted as a "state network", is shown in Figure 1-4. We use the pair (j, t) to identify the network state based on which the decision is made, where j is a node and t is a time point. The traveler starts from $(1, 0)$, and the decision is to go to node 2. At node 2, two situations $(2, 2)$ or $(2, 4)$ are possible. With $(2, 2)$, the traveler chooses link a, and could end up at either of the following two situations: $(3, 4)$ or $(3, 6)$. Similarly with $(2, 4)$, the traveler chooses link b, and could end up at two situations: $(3, 6)$ and $(3, 8)$.

From the above two examples, we can see that a routing policy generally involves more than one path. Which path is to be taken depends on the network conditions, i.e. link travel time realizations and/or the arrival times. When one is at a given node, the least expected travel time path implicitly does not exploit the possible information collected during the trip, and thus is generally less effective than a best routing policy. The value of information is either due to the statistical dependency or or due to the time dependency of stochastic link travel times. One can make various assumptions about the statistical dependency of link travel times and the degree of knowledge one would have about available link travel times. These factors lead to

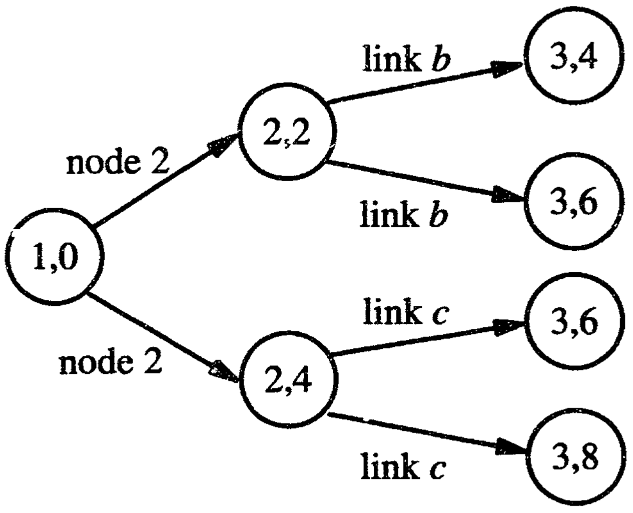


Figure 1-4: The Best Routing Policy for the Example in Figure 1-3

numerous variants of the BRP problem in a STD network. A limited number of these variants have been studied in the literature, and some will be explored in this thesis for the first time.

1.3 Thesis Contributions

The contributions of the thesis to the knowledge base of routing problems in stochastic time-dependent networks are summarized as follows:

1. The concept of routing policies is well developed. The study of routing policies in STD networks in the literature has been restricted to policies based on arrival times on decision nodes. This thesis recognizes the role of information in routing decision making, and includes information as an integral part of a routing policy.

2. The first framework for BRP problems in STD networks is established. Various assumptions have been made in the literature to define routing problems in stochastic networks. These studies, however are ad hoc. This thesis identifies

the similarities among these variants and establishes a framework for a unifying understanding of the problem.

3. A comprehensive taxonomy of the BRP problems in STD networks is provided. The taxonomy is based on information access and network statistical dependency. It contains all variants in the literature and can lead to new variants suitable to various applications.

4. Statistical dependency of link travel times is modeled and a solution algorithm is designed. Traffic networks are generally statistically dependent both link-wise and time-wise, yet no papers in the literature have addressed this problem in STD networks.

5. The importance of designing good approximation algorithms for BRP problems is identified. This thesis provides four possible approximation algorithms and studies their effectiveness both theoretically and computationally. These studies are the first step to designing time-effecient routing policy algorithms for real-time traffic applications.

The contributions of the thesis to the knowledge base of dynamic traffic assignment are summarized as follows:

1. The first DTA model that works with general link travel time distributions is developed. Most current dynamic traffic assignment models work with deterministic networks. Some others assume specific forms of distribution (e.g. normal distributions) for link travel times. This thesis thus allows for a better representation of stochasticity in traffic modeling.

2. The first policy-based DTA model is established. Results from a policy-based DTA model and a path-based DTA model are compared and shown to be different. This suggests further study on implications of routing policies in DTA models.

3. The first DTA model that outputs link travel time distributions is built. This property of the DTA model allows for a richer representation of traffic. Better traffic managment decisions are then possible based on this richer representation.

1.4 Thesis Organization

The thesis is organized as follows. We study the stochastic routing problem first. We propose exact algorithms and approximations. We then design a policy-based stochastic dynamic traffic assignment model.

In Chapter 2, we study the best routing policy problem in a stochastic time-dependent network. We give a framework of the problem which includes a general description of a STD network, the decision process, the problem statement and the optimality conditions. We then present a comprehensive taxonomy based on assumptions of the network statistical dependency and information access. A discussion of nearly each variant is given. Specifically two variants are studied in details. The first one is the no-information variant which is easy to understand and can be solved in polynomial times. We give the formulation, an algorithm and computational tests for this variant. Two algorithms exist for this variant and we make comparison between them both theoretically and computationally. The second variant studied in detail is the perfect-on-line information variant, which is pertinent to transportation applications. This variant has never been studied before in the literature. We give a formulation, an algorithm, and results from computational tests. The complexity analysis shows that the algorithm for the second variant can be prohibitively time-consuming. Therefore good approximations are also developed.

Chapter 3 studies approximations to the second variant studied in Chapter 2. Four approximations are presented with analysis on their efficiency and effectiveness. This analysis is done both theoretically and computationally. The computational tests are not comprehensive, but they provide insights into the performance of approximations. Other approximations are suggested, however without computational tests.

In Chapter 4, we develop a dynamic traffic assignment model. The model fundamentally differs from traditional ones, in the sense that it is based on routing policies, rather than paths, to explicitly model the stochasticity in a dynamic traffic context. It tries to achieve more accurate results than traditional deterministic DTA models, as it models link travel times as stochastic random variables. An illustrative example is used to show the unique characteristics of a policy-based traffic assignment model. We then present a users' policy choice model and a dynamic traffic network loading model. All models are based on routing policies. A stochastic DTA heuristic is then proposed based on the routing model, the user's policy choice model and the dynamic network loading model.

Chapter 2

Best Routing Policy Problems in Stochastic Time-Dependent Networks

In this chapter, we study best routing policy problems in stochastic time-dependent networks. We first provide a literature review on a broad range of routing problems in networks. We then establish a framework, in order to provide a unified view toward this problem, considering the large variety of variants already in the literature and the numerous possibilities of new variants. This framework includes a general description of a stochastic time-dependent network, the decision process in a stochastic time-dependent network, the minimization problem, and the optimality conditions. Following this somewhat abstract framework, we give comprehensive taxonomy based on two criteria: the network statistically dependency and the information access. We discuss the variants within the taxonomy, and pay special attention to the variants already in the literature to see how they fit in the framework. This discussion may still seem abstract, as no algorithms are given at this point. We suggest that the reader come back to this part after he/she finishes the following algorithmic parts. We study two variants in details after the framework and taxonomy, The first is the no-information variant which is not so realistic in traffic settings, but has a very encouraging running time and is relatively easy to understand. We study it first to provide some knowledge preparation for the more complicated variant. Next we study the perfect-online-information variant where the dependency of traffic network is considered. Both variants are studied in a formal way: the formulation, algorithm, implementation, complexity analysis, and computational tests are presented in sequence. The study of the perfect-online-information variant also suggest the need to design good approximations to the exact algorithm, which is the topic of the next chapter.

2.1 Literature Review

The routing problem in networks has been an important and well researched topic for a long time. We first give a brief introduction of the shortest path problem in deterministic networks, including the well developed static shortest path (SSP) problem and the dynamic shortest path problem. This will be useful to the study of routing problems in stochastic networks. We then proceed to stochastic networks. There are various assumptions about how a stochastic network is defined, and this results in a variety of variants of the BRP problem. Most of the problem variants studied in the literature assume that the underlying network is static (not dependent on time). Some other variants studied in the literature work with a special case of dynamic stochastic networks. They do not represent time explicitly. These variants can be viewed as the infinite horizon version of the BRP problem in a STD network. A limited number of papers studied the BRP problem in a STD network with specific assumptions. A comprehensive study of the problem is not available in the literature.

2.1.1 Deterministic Routing Problems

Compared to routing in STD networks, the classical static shortest path (SSP) problem has been more extensively studied. Let $G(N, A)$ be a network, where N is the set of nodes and A is the set of links. Each link (i, j) has a cost $c(i, j)$. The SSP is to find a shortest path for a source node s and a destination node d. Dijkstra's algorithm is the most commonly used algorithm to solve the shortest path problem for networks with non-negative costs. Various implementations of Dijkstra's algorithm exist. The most straightforward one is based on the data structure of array and has a running time of $O(n^2)$, where n is the number of nodes. The implementation using a binary heap can achieve a running time of $O(m \ln n)$, where m is the number of arcs. If the network has negative arc cost, more sophisticated algorithms (such as the label-correcting algorithms) are needed. These algorithms basically check whether the optimality condition $d(i) + c(i,j) \geq d(j), \forall (i,j) \in A$, where $d(i)$ is the distance label for node i, are satisfied. They make necessary changes by changing distance labels until no arec viloates this condition. A first-in-first-out (FIFO) queue implementation of the label correcting algorithm has a running time of $O(mn)$.

The dynamic shortest path problem becomes interesting when modeling of the transportation system with large variability in trvel times as a function of time is required. Let $G(N, A, T)$ be a dynamic network. T is the set of time periods. At each time period t, each link (i,j) has a cost $c_{ij}^{t} > 0$. The dynamic shortest path problem is to find a shortest path from a given source node s to a destination node d for a given departure time t at node s. We define a link (i,j) as FIFO, iff $t_1 + c_{ij}^{t_1} \geq t_2 + c_{ij}^{t_2}, \forall t_1, t_2 \in T$ and $t_1 > t_2$. A network is FIFO, iff all links are FIFO. When the dynamic network is FIFO, we can apply a Dijkstra-like algorithm to solve the dynamic shortest path problem. When the network is non-FIFO, generally a label-correcting-like algorithm would be able to solve the problem. Chabini [9] presents an algorithm DOT with an optimal running time $\theta(SSP + mK + nK)$ to the dynamic shortest path problem with positive travel times from all nodes at all departure time to one destination node, where SSP is the running time of a static shortest path algorithm and K is the number of time periods. Algorithm DOT is optimal in the sense that no algorithm with better theoretical running time exists. Algorithm DOT sets the labels in decreasing order of time, based on the fact that the distance label of a node at a given time can only be updated by labels of later times. This algorithm is the base of the algorithms we develop for the stochastic routing problem.

2.1.2 Routing in Stochastic Static Networks

Loosely speaking, a stochastic network is a network where the link travel times are random variables with some *a priori* distributions. If the underlying network is assumed to be static (non-time-dependent), the link travel times remain unchanged after they are revealed to the travelers. While in a time-dependent network, the travel time of every link at every time period is an individual random variable, so travel times revealed at different time periods could be different. The study of BRP problems in static networks is useful to the study of its time-dependent counterpart.

Andreatta and Romeo [2] study the problem in a static network where the topology is stochastic. A stochastic topology is defined by a deterministic set of nodes N and a random set of links $A \in N \times N$. Each possible topology is associated with a positive probability. The decision maker (DM) can learn whether a link is active or not once he/she reaches the node from which the link emanates from. The DM can take recourse once he/she finds out the next link is inactive. The notion "stochastic shortest path" is used, yet actually a routing policy problem is studied. The path without recourse actions in a routing policy (i.e. the path composed solely of nodes representing "active" scenarios in the state network of a routing policy) is used to denote that policy, so a stochastic shortest path in this paper is actually a least expected cost routing policy. Andreatta and Romeo [2] proves four facts about a stochastic shortest path that are different from those about a deterministic shortest path. A stochastic dynamic programming formulation of the problem is provided, with the definition of "information state" which gives the active/inactive links of the network revealed to the decision maker so far and based which the recourse decision is made. It is pointed out the complexity of the algorithm can grow exponentially with the number of links. Therefore a restricted version of the problem is studied and it is shown polynomial algorithms exist for this particular case.

Polychronopoulos and Tsitsiklis [25] extend the work of [2]. They study the problem both in networks with strongly dependent link travel times and in networks with independent link travel times. For the dependent case, a joint distribution of link travel times is used to represent the stochastic network. We can see that the stochastic topology in [2] is actually one special form of joint distribution of link travel times. It is assumed that the travel time realizations of outgoing links of a given node is known and remembered by the traveler once he/she arrives at this node, and the realizations remain unchanged afterward. As the traveler moves on the network from the origin to the destination, more link travel time realizations are learned, and the network becomes closer to a deterministic one. The concept of information set is introduced to represent the traveler's knowledge about the network. An information set is composed of joint realizations that are consistent with the revealed link travel times so far. When the information set becomes a singleton, the network becomes deterministic. A dynamic programming approach is presented where the stage of dynamic programming is labeled by the cardinality of the information set, starting from the smallest. Some of the main concepts in the present paper originate from [25]. A similar approach is designed for the independent case, with changes in the manner in which the information set is defined. The algorithms, however, have exponential running times: the algorithm for the dependent case has running time exponential in the number of joint realizations, and the algorithm for the independent case exponential in the number of links. It is proved that the problem with dependent link travel times is NP-complete, and that with independent link travel times is #P-hard. Some heuristics are given and the relationship between results from heuristics and exact algorithms are studied.

Cheung [13] studies the problem with the same independent network assumptions as those in [25], except the assumption that two visits to the same node result in different realizations of outgoing link travel times. This assumption actually make ambiguous the statement that the network is static, as the same link can take different travel times at different time periods, although the distribution is the same. An approach that mimics the classical label-correcting algorithm is presented. Computational tests are carried out to compare different implementations of the label-correcting approach.

2.1.3 Routing in Stochastic Time-Dependent Networks

Hall [17] studies for the first time the time-dependent version of the best routing policy problem. The problem is studied within the context of transit networks. It is shown that in a stochastic time-dependent network, adaptive route choices (routing policies) are more effective than simple paths. A dynamic programming approach is provided, where the stages of the dynamic program are the number of links from the destination node. An upper bound k on the number of stages is specified, and it is stated that when k is sufficiently large, the solution should be very close to the optimum. The recurrence equations for the dynamic program are given and it is implicitly assumed in the equations that routing policies are based only on arrival times at decision nodes. We note that with this implicit assumption, k can be set to be the sum of number of time periods and number of nodes to guarantee the optimality of the solution.

The assumption that routing policies only depend on arrival times at decision nodes is also made by Chabini [10]. A dynamic programming algorithm where the stage of the program is the time period t is developed, based on the concept of decreasing order of time that is also used in developing Algorithm DOT [9]. This formulation of the problem enables an optimal algorithm DOT-S with a complexity of $\theta(SSP + nK + mKQ)$, where Q is the maximum number of realizations for a single link travel time distribution. This algorithm is optimal in the sense that no algorithms with better theoretical complexity exist. The algorithm is extended to solve the minimum expected travel cost routing policy problem with minor changes. Computational tests are carried out to study the running times of Algorithm DOT-S and the label-correcting algorithm developed in [20]. It is concluded that algorithm DOT-S is computationally efficient both in theory and in practice.

Miller-Hooks and Mahmassani [20] study the BRP problem assuming time-wise and link-wise statistically independent link travel time random variables. This assumption leads to routing policies based only on arrival times at decisions nodes. A label-correcting algorithm is developed to solve the problem. The label-correcting algorithm has a rather high worst-case running time, but its practical performance proves to be good. Miller-Hooks [21] compares the label-correcting algorithm presented in [20] and the dynamic programming algorithm working in decreasing order of time [10] in both sparse transportation networks and dense telecommunication data networks. It is shown that the label-correcting algorithm has an empirical running time much better than its worst-case theoretical complexity. It is also concluded that in dense networks, the label-correcting algorithm is more computationally efficient than algorithm DOT-S. This conclusion is somehow against the theoretical analysis, and computational tests are carried out in this thesis to study the problem.

We also make a brief literature review on the least expected time path problem in a STD network, as it is closely related to the BRP problem. Fu and Rilett [16] model link travel times as a continuous-time stochastic process. It is assumed that travel times on individual links at a particular point in time are statistically independent, and the correlation between link travel times are modeled through the time-dependency of link travel time distributions. Relationships between the mean and variance of the travel time of a given path and the mean and variance of link travel times on that path are identified. A heuristic is designed in recognition of the computational intractability of the problem. Miller-Hooks and Mahmassani [20] study the least expected time path problem under the same assumptions for the BRP problem. They establish a dominance rule for paths in STD networks and design a label-correcting-like algorithm. The worst-case complexity of the algorithm is exponential as a function of the network size, but computational tests on sparse transportation networks show the actual performance is practically linear with respect to the network size.

Some researchers studied the BRP problem variants with stationary Markovian link costs and these variants can be viewed as an infinite horizon version of the dynamic BRP problem. Polychronopoulos [24] assumes global information access and defines a combination of travel times of all links as a state. It is further assumed that the transition probability matrix is available, and that the occurrence of transition in unit time is related to the network conditions. A dynamic programming formulation of the problem is suggested and it is claimed that any standard Markov decision algorithm can solve the problem. Psaraftis and Tsitsiklis [26] assume travel times of outgoing links of a given node are functions of condition at this node, which evolves as a Markovian chain. Markovian chains at different nodes are assumed to be independent and the network is acyclic. Vehicles can wait at a node (at a cost) in anticipation of more favorable arc cost. Three different types of algorithms are developed to solve the case of single arc network: successive approximation (SA), policy iteration (PI), and parametric linear programming. A dynamic programming approach is then developed, making use of the algorithm for a single arc. The algorithm is shown to be polynomial, due to the assumptions of acyclic networks and stationary Markovian costs independent across nodes.

2.2 Framework and Taxonomy

2.2.1 Framework

Through the literature review, we find that there is not a formal definition of the routing problem in a stochastic time-dependent network (even not for the non-time-dependent network). Various assumptions are made to define a stochastic network and to define how the realizations of the stochastic network are revealed to the travelers (decision makers). For example, in [2], the topology of the network is stochastic; in [25], the whole static network is described by joint distribution of link travel costs; in [24] and [26], the link costs evolve as Markov chains; in [17], [10] and [20], time-dependent networks are described by marginal distributions of link travel times. As for the revealing of the stochastic network, some assume that one learns the realization of a link travel cost once he/she arrives at the node from which the link emanates from [2] [25] [13], while most papers do not state explicitly how travelers learn about the network conditions as their formulations of the problem can be validated solely by their assumptions of statistical Independence of the network [26] [10] [20]. Yet we can have these various descriptions and assumptions generalized. We also realize that the routing process in a stochastic network is merely a mapping from some knowledge of the network to a decision, and what knowledge is available and/or useful depends on specific assumptions about the network and the information access, as shown in different papers in the literature. A general set of optimality conditions is then possible with the formal definitions of the problem.

We establish the framework to provide a unified view to toward the best routing policy problem in a stochastic time-dependent network. We will be able to see the connections among various variants in the literature with the aid of the framework, and to gain insight of generating new variants that are required by specific applications. The general optimality conditions can provide a general way of designing solution algorithms for variants of the problem.

The Network

Let $G = (N, A, T, P)$ be a **stochastic time-dependent network.** N is the set of nodes and A is the set of links. The number of nodes and links are denoted respectively as $|N| = n$ and $|A| = m$. The network has a destination node d. T is the set of time periods $\{0, 1, ..., K-1\}$. Travel time of each link (j, k) at each time period t is a random variable $C_{jk,t}$ with discrete, non-negative and integral realizations. Beyond time period $K-1$, travel times are static and deterministic, i.e. travel times of link (j, k) at any time $t \geq K-1$ is equal to $c_{jk,K-1}$. In this thesis, we only study the problem of finding least expected travel *time* routing policies, but the study can be easily extended to the problem of finding least expected travel *cost* routing policies. For this reason, we do not define link costs here.

P is the probabilistic description of link travel times. Different descriptions exist because of different assumptions about network statistics. The most general one is in the form of joint probability distribution of all the link travel time random variables, which is described next. Let $P = \{v_1, v_2, ..., v_R\}$ be the set of possible joint realizations of link travel times, for all links and all time periods. The rth realization has a probability p_r, and $\sum_{r=1}^{R} p_r = 1$. $c^r_{jk,t}$ is the travel time of link (j, k) at time t in the rth realization. Note that we assume the underlying topology of the network is deterministic, as a network with stochastic topology can be transformed into a network with deterministic topology where a blocked (or missing/inactive) link is modeled by setting its travel time to infinity (or computationally, a very large positive number).

We will use an example to show how the joint distribution description works. Figure 2-1 shows a small network with three nodes, three links and the number of time periods is 3. The values of the travel time realizations are in Table 2.1. Each of the eight realizations has a probability of 0.125. The network is designed to be very small to make the understanding of the concept easier.

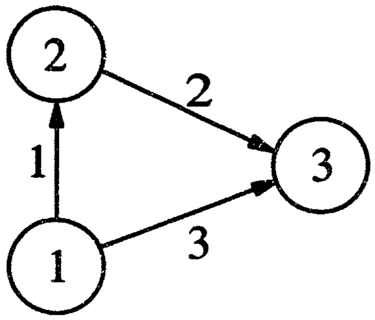


Figure 2-1: A Small Network

Time	Link	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
0	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1
	3	1	1	1	4	4	4	3	3
1	1	1	1	1	1	1	1	1	1
	2	2	2	1	2	2	1	2	1
	3	3	3	2	2	2	1	3	2
$t \geq 2$	1	1	1	2	1	1	1	2	2
	2	1	2	1	1	1	1	2	1
	3	3	2	2	3	4	3	5	2

Table 2.1: Joint Realizations for the Small Network

The joint realization description of the network statistics can be specialized to other descriptions, depending on the assumptions about the network statistics. If all the link travel time random variables are statistically independent, both link-wise and time-wise, we could still use the joint realization description. However, it would be much more efficient if we only keep the marginal distributions of link travel times. If we assume all links can be divided into several groups and link travel times in a given group are independent of link travel times outside the group, we need only the joint distribution of link travel times in each group. The same grouping can also be done along the time dimension, or along both the time dimension and link dimension.

The Decision Process

Throughout the thesis, we assume the traveler knows *a priori* the probabilistic description P of the network. Assume the traveler can make decisions only at nodes. The decision is what node k to take next (no waiting is allowed), based on the *current state* $x = \{j, t, I\}$, where j is the *current-node*, t is the *current-time*, and I is the *current-information*. Current-information I contains links whose travel time realizations are useful in making inferences about future link travel times. It represents the traveler's knowledge about the network conditions. This knowledge could be dependent on time, location of the traveler, mode of the transportation, etc. Current-information I therefore should be regarded as $I(j, t)$, but we usually use only I to denote it as I is always associated with a state where j and t are well defined. More discussion about current-information can be found in the next subsection about taxonomy. An ideal case is that travelers have perfect information about the whole network, but generally the information is local, as shown in the example of Figure 1-2, where one learns the travel time realization of a link when he/she arrives at the node from which the link emanates from. In this example, the current-information would be the combination of link travel time realizations of link a and link c. The decision at node 1 can then be described as: when current state is $\{1, t, (2, 1)\}$, take node 3 next; when current state is $\{1, t, (1, 3)\}$, take node 2 next, for all t. Note that "current-information" is one component of a current state and refers to link travel time realizations based on which the current decision is made, while a reference to "information" alone is in the general sense. One can be in many different states traveling in the stochastic time-dependent network, and we define a **routing policy** $\mu(x)$ as a mapping from states to decisions (next nodes specifically in networks).

This definition indicates that the routing decision in a stochastic time-dependent network is far from being set *a priori*. Rather, it is closely related to the network conditions, and this notion is critical in any ITS applications.

We look ahead after making the decision at the current state. We do assume that the realization of the decision is certain, i.e. the traveler will end up arriving at node k if he/she chooses it. The next state $y = \{k, t', I'\}$ the traveler will occupy is uncertain, i.e. t' and I' are random variables. The travel time of link (j, k) at time t conditional on I could be uncertain, resulting in an uncertain arrival time t' at node k. The next current-information I' is also uncertain, as t' itself is uncertain. Even if t' is certain, link travel time realizations from t to t' could take multiple values, as the network is still stochastic to the traveler at current state. However, for a given current state and a given decision, probabilities of all possible next states can be evaluated from the network statistics P.

Define a *state chain* $\{x_0, x_1, ..., x_S\}$ as the series of states a traveler experiences during the trip, where x_S is a state with the destination node d as its current-node.

Current-nodes of a state chain form a path, and S is the number of links in the path. With a given initial state x_0 and a routing policy μ, one could experience multiple state chains. For example, the routing policies in Figure 1-2 and Figure 1-3 involve more than one path. As stated in to the visualization of a routing policy in Figure 1-4, a routing policy with an initial state can be visualized as a **state network**. In this state network, a node is a state and outgoing links of a node is the decision based on that state. The succeeding nodes stand for the possible next states the traveler will be in.

Denote this set of possible state chains for a given initial state x_0 and a given policy μ as $M(x_0, \mu)$, then the state network is a representation of $M(x_0, \mu)$.

We use the small network example in Figure 2-1 and Table 2.1 to show the decision process. We assume the traveler have knowledge of all the link travel time realizations up to the current-time, regardless of his/her current-node. This assumption implies the following: at time 0, the traveler knows what travel time values of link 1, link 2 and link 3 take as of time 0; at time 1, the traveler knows what link travel time values of link 1, link 2 and link 3 take as of time 0 and time 1; at time 2, the traveler knows what link travel time values of link 1, link 2 and link 3 take as of time 0, time 1 and time 2; etc. Therefore the current-information I at time t would be one of the joint realizations of $C_{1,0}$, $C_{2,0}$, $C_{3,0}$, ..., $C_{1,t}$, $C_{2,t}$, and $C_{3,t}$. Note that we use one single link number to denote a link rather than a pair of node numbers, for the sake of simplicity.

We seek to travel from node 1 to node 3. As there is no choice at node 2 or node 3, our focus is at the choice at node 1. A naive threshold routing policy would be: for all time t, take node 3 if travel time of link 3 at current-time is less than 3, and take node 2 otherwise. This routing policy can be understood as follows. We can view node 1 as the traveler's home, and node 3 as the traveler's work place. Link 3 is an artery. Link 2 is part of a freeway, and link 1 is a ramp to the freeway. The traveler make some observation at the home location. If he/she finds out that it takes less than 3 unit time to travel on the artery, he/she is pretty sure he/she will be better off to take it. Otherwise, he/she will conclude that the artery is congested, and he/she will just take the freeway. Under all the above specifications, the routing policy with an initial state $\{1, 0, (1, 1, 4)\}$ is shown in Figure 2-2.

Let us now go through the state network step by step. The initial state is $\{1, 0, (1, 1, 4)\}$. We can see from the joint realization table that the network could be in v_4, v_5, or v_6. As travel time of link 3 at time 0 is greater than 3, the traveler chooses node 2 as the next node. When he/she arrives at node 2, he/she could be in two possible states. One is $y = \{2, t', I'\} = \{2, 1, (1, 1, 4, 1, 2, 2)\}$ as represented by the upper one of the two succeeding nodes of node 2 in the state network. The other is $y = \{2, t', I'\} = \{2, 1, (1, 1, 4, 1, 1, 1)\}$ as represented by the lower one of the two

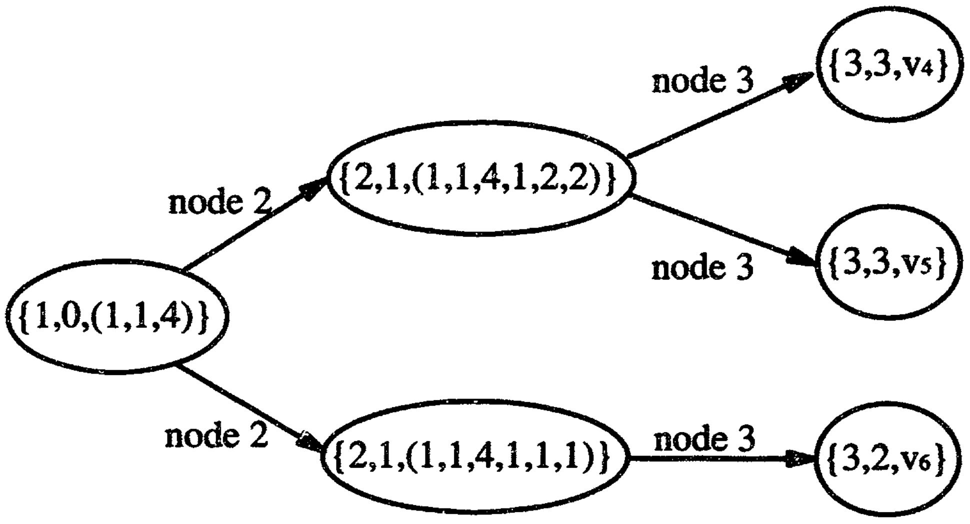


Figure 2-2: The Decision Tree of the Naive Routing Policy

succeeding nodes of node 2. The only choice at node 2 is node 3, and the traveler arrives at the destination (node 3). However, the ending states could be different. From the state $\{2, 1, (1, 1, 4, 1, 2, 2)\}$, the traveler could end up at state $\{3, 3, v_4\}$ or state $\{3, 3, v_5\}$. From the state $\{2, 1, (1, 1, 4, 1, 1, 1)\}$, the traveler could end up at state $\{3, 2, v_6\}$.

There are altogether three state chains in this state network. Note that the arrival times at the destination are different for different state chains. For all state chains, the arrival time at node 2 is 1, as the travel time of link 1 at time 0 is 1. For the upper two state chains, however, the link travel time of link 2 at time 1 is 2, so the arrival time at node 3 is $3(= 1 + 2)$. For the lower state chain, the link travel time of link 2 at time 1 is 1, so the arrival time at node 3 is $2(= 1 + 1)$.

We see that for a given routing policy and a given initial state, the O-D travel time is a random variable. For example, the O-D travel time as shown in the state chain of Figure 2-2 is a random variable with two possible realizations: 2 and 3. The probability that the O-D travel time is realized as 3 is the probability the state chain is realized as the upper two chains, which is the probability that link travel time realizations for all links at time 1 is $(1, 2, 2)$. Note that this probability should be evaluated conditional on the fact the link travel time realizations for all links at time 0 is $(1,1,4)$. Therefore the probability in question is

$$\frac{p_4+p_5}{p_4+p_5+p_6}=\frac{0.125+0.125}{0.125+0.125+0.125}=\frac{2}{3}.$$

Similarly, the probability that the O-D travel time is realized as 2 is 1/3. Therefore the expected O-D travel time for the routing policy with the given initial state as in Figure 2-2 is $3\times 2/3+2\times 1/3=8/3$, and the variance is $(3-8/3)^2\times 2/3+(2-8/3)^2\times 1/3=5/27$.

The Minimization Problem

In traffic applications, we want to reach the destination in an optimal way. Since link travel times are random variables, there exist multiple criteria on what optimal travel times are. Usually the primary concern of routing is the expected travel times from origins to destinations, i.e. a routing policy with less expected travel time is a better one. However, the variances of O-D travel time random variables are also important. Depending on the traveler's attitude toward risk and the type of trips, he/she will make trade-offs between expected travel times and variances. A good routing model should be able to handle this trade-off.

The expected travel time is used as the only criterion of optimization at the time being, and the risk-taking behavior will be modeled in Chapter 4. Define t_x as the current-time of state x, and $E[Z]$ as the expectation of random variable Z. The **best routing policy problem in a stochastic time-dependent network** with one destination node d is to find μ^*, such that

$$\mu^*=\arg\min_{\mu}\{E_{\{x_0,x_1,\dots,x_S\}\in M(x_0,\mu)}[t_{x_S}-t_{x_0}]\},\quad \forall x_0 \tag{2.1}$$

The random variable to be taken expectation is $t_{x_S}-t_{x_0}$, the travel time from the origin as defined in the initial state x_0 to the destination node d for a given routing policy μ. The expectation is taken over all possible state chains, $M(x_0,\mu)$. The minimum is taken over all routing policies. Note a best routing policy is optimal for all initial states, not just for a specific initial state.

We can compare the best routing policy with a shortest path tree in the deterministic and static all-to-one shortest path problem. The classical all-to-one shortest path problem is to find the shortest paths from all nodes to one destination node in a static and deterministic network. The result is a directed in-tree rooted at the destination node. The shortest path from any node j to d is the path from j to d in the shortest path tree. The shortest path tree can be viewed as a specialized routing policy, where there is only one possible state for a given node and the decision (next node) for that state is the successor node in the shortest path tree. In the classical all-to-one shortest path problem, *all* stands for "all nodes", while in the best routing policy problem in a STD network, we have an implicit *all* standing for "all times" and "all current-information" as well as all nodes. A counterpart of the shortest path tree in the best routing policy problem would be the union of state networks for all the possible states. There is no guarantee that the state network union is acyclic or connected, however, as opposed to the shortest path tree.

The Optimality Condition

Let $e_\mu(x)$ denote the expected travel time to the destination node d when the initial state is x and the routing policy μ is applied. Define $A(j)$ as the set of adjacent nodes of node j, $C_{jk,t}|I$ as a travel time random variable of link (j,k) at time t conditional on current-information I, and $I'|I$ as a current-information random variable at the next node k and at time $t + C_{jk,t}|I$. For $\forall j \in N - \{d\}, \forall t \in T, \forall I$ that is possible at node j and at time t, $e_{\mu^*}(x)$ and μ^* are optimal if and only if they are solutions of the following system of equations:

$$e_{\mu^*}(j,t,I) = \min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + E_{I'}[e_{\mu^*}(k, t + C_{jk,t}, I')]|I]\} \tag{2.2}$$

$$\mu^*(j,t,I) = \arg\min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + E_{I'}[e_{\mu^*}(k, t + C_{jk,t}, I')]|I]\} \tag{2.3}$$

with the boundary conditions: $e_{\mu^*}(d,t,I) = 0, \mu^*(d,t,I) = d, \forall t \in T, \forall t \in T, \forall I$ that is possible at node d and at time t. Since $I'|I$ is dependent on $C_{jk,t}|I$, we first take the expectation over $I'|I$ with a given realization of $C_{jk,t}|I$ and then take the expectation over $C_{jk,t}|I$. Note that we assume the realization of the decision is deterministic, i.e. the traveler will end up at node k if he/she chooses node k as his/her next node. Croucher [14] studies the problem where the realization of the decision itself is stochastic. We do not discuss this case, as our original initiative in studying the BRP problem is for traffic applications where this case rarely arises.

The proof of the optimality conditions is similar to the proof of Proposition 7.2.1 in [4]. The problem in [4] is denoted as a stochastic shortest path problem and is viewed as an infinite horizon dynamic programming problem. The proof provided in [4] uses only the node number as a state, yet we can simply replace the state by $\{j, t, I\}$ and the proof becomes valid for our case.

2.2.2 Taxonomy

In this subsection, we give taxonomy of the best routing policy problem in a STD network. There are four major objectives of providing taxonomy:

- To make the abstract framework concrete and applicable to traffic context
- To show the variety of the best routing policy problems
- To study the role of information in a stochastic routing context
- To gain insight of the complexity of the problem

The framework is abstract in the sense that no concrete form of current-information I is specified. Current-information depends on two factors: network statistical dependency defined as the statistical dependency of link travel time random variables, and information access defined as the link travel time realizations that are available to the travelers at any given time and given node. The taxonomy of the BRP problem is therefore along these two dimensions. We will see that depending on the assumptions about the two factors, we can have a large variety of the BRP problem variants. Some of them are just for the purpose of theoretical analysis, while others are realistic in traffic context. Specifically we can see the role of information in stochastic routing. In fact, ITS applications rely to a large extent on the acquisition and processing of information on traffic conditions, therefore the study of the role of information is needed. During the discussion of each variant, we give a brief overview on the complexity of the BRP problem and show how the complexity varies from variant to variant.

Taxonomy

Network statistical dependency is characterized by link-wise and time-wise statistical dependencies of link travel times. At one extreme, all the link travel time random variables are independent, both link-wise and time-wise. At the other extreme, all the link travel time random variables are strongly dependent. There are numerous cases in between these two extremes, and we denote them as partial statistical dependency.

Information access has the following four categories:

- Perfect *a priori* information
- Perfect on-line information
- Partial on-line information
- No information

Travelers with perfect *a priori* information have knowledge of the realizations of all link travel time random variables before the trip. Travelers with perfect on-line information have knowledge of the realizations of all link travel times up to current time period. Travelers with partial on-line information only have knowledge of part of the link travel time realizations and the restrictions in on-line information can be either temporal, spatial or both. Travelers with no information have no knowledge of any of the realizations and the only knowledge they have about the current state is the current-node and current-time. Table 2.2 gives a possible taxonomy along the two dimensions.

	Perfect *a priori* Information	No Information	Perfect On-line Information	Partial On-line Information
No link-wise and no time-wise dependency	WS (wait-and-see)	NI	Group 1	
Strong dependency			Group 2	Group 3
Partial dependency				

Table 2.2: Taxonomy of the BRP Problem

Discussion of Taxonomy

In the discussion of the variants listed in Table 2.2, we focus on the specification of current-information for each variant and the resulting implications for algorithm design. A general rule in determining current-information is as follows: information access determines which link travel times have the potential to be included in current-information, while network statistical dependency determines whether all the available link travel time realizations are necessary. The unnecessary link travel times can be eliminated so that the dimension of current-information is minimized. For example, assume we are equipped with the most advanced traffic information system so that we know the realizations of all link travel times up to current time (i.e. perfect on-line information). Presumably we hope we can make use of all the available information. However, assume all the link travel time random variables are statistically independent, implying that knowledge about one link cannot help infer about any other links, then none of the information is useful and the current-information is actually an empty set. This rather extreme case shows how the two factors act together to determine a current-information, and we will see more in the following discussion.

The WS variant has perfect *a priori* information and any kind of network statistical dependency. We borrow a term from stochastic programming to denote the variant as WS (wait-and-see). In WS, the current-information I includes all the link travel times at all time periods, so travelers can know the network deterministically *a priori*. To put it in mathematical phases, in a network $G = (N, A, T, P)$ as defined in subsection 2.2.1, current-information $I = A \times T$ and the traveler knows *a priori* which joint realization of link travel times, $v_1, v_2, \ldots$ or v_R, the network will take. This variant is not realistic, as in reality the future is always uncertain to some extent. It is discussed here, since for a network with a given type of statistical dependency, WS variant gives a solution lower bound for all other variants of the BRP problem. It can be used as a benchmark in the robustness analysis of solutions to other variants.

Under perfect *a priori* information, the BRP problem reduces to multiple deterministic dynamic shortest path problems, each of which works on a deterministic network defined by one of the R joint realizations $v_1, v_2, \ldots, v_R$. Since we are working only on deterministic networks, network statistical dependency does not make any difference in algorithm design. Algorithm DOT [9] with a running time of $\theta(SSP + nK + mK)$ for all-to-one shortest path problem, where SSP is the running time of a classical static shortest path algorithm, can be used to solve the individual deterministic dynamic shortest path problems. The complexity of the WS variant is then $\theta(R \times (SSP + nK + mK))$.

The No-Information variant is the other extreme case when no information (NI) is available. The lack of information prevents travelers from being able to make any useful inferences about future network conditions. The current-information is an empty set at any point in space and time, and decisions depend only on current-node and current-time. This is true for any kind of statistical dependency, which is another example besides the WS variant showing the role of information in defining a BRP problem. As the current-information is an empty set, we can simply remove it from the current-state, and the optimality conditions in subsection 2.2.1 reduce to

$$e_{\mu^*}(j,t) = \min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.4}$$

$$\mu^*(j,t) = \arg\min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.5}$$

Several algorithms have appeared in the literature to solve this variant [17] [10] [20], yet no explicit discussion of the role of information is provided. It is sometimes papers that link travel times are statistically independent so as to obtain the optimality conditions as shown above. However, the assumption of statistical independence is neither sufficient nor necessary to validate (2.4) and (2.5). It is not necessary, because variants with statistically *dependent* link travel times and no information can have this formulation. It is not sufficient, because if the realizations of outgoing link travel times at current-time is available in a statistically *independent* network, the current-information is no longer an empty set and thus the above optimality conditions of the problem is no longer valid.

Algorithm DOT-S [10] has an optimal running time of $\theta(SSP + nKQ + mKQ)$ for the NI variant, where Q is the maximum number of realizations of a single link travel time, in the sense that no algorithms with less theoretical complexity exist. In Section 2.3, the solution algorithms for the NI variant are extensively discussed and computational tests are presented.

The Independent variants. In the rest of the section, we discuss variants with some online information access. The above discussion shows that sometimes information access alone can determine the current-information, as in the case of WS and NI. On the other hand, network statistical dependency sometimes can play a very important role in determining the current-information. This can be shown by the variants in Group 1 with statistically independent link travel times. First of all, the knowledge about the adjacent links of the current-node at the current-time is useful, as they are explicitly included in the optimality conditions by all means. Any other link travel time realizations, however, cannot contribute to the decision making. Define $\delta(j)$ as the adjacent links of node j. This fact then is stated formally as follows:

Theorem For a given network $G = (N, A, T, P)$ as defined in subsection 2.2.1 where the link travel time random variables $C_{jk,t}$ are statistically independent, $\forall (j,k) \in A, \forall t \in T$, define two types of current-information:

1. $I_1 = A \times \{0, 1, ..., t\}, \quad \forall t \in T$
2. $I_2 = \delta(j) \times \{t\}, \quad \forall j \in N, \forall t \in T$

Let μ_1^* and μ_2^* be the best routing policy respectively with the first and second definition of current-information. If travel time realizations of $\delta(j) \times \{t\}$ are the same in $I_1(t)$ and $I_2(j,t)$, we have

$$e_{\mu_1^*}(j,t,I_1) = e_{\mu_2^*}(j,t,I_2), \forall I_1, I_2, \forall j \in N, \forall t \in T.$$

Proof: We use induction on time t to prove the theorem. Since travel time realizations of $\delta(j) \times \{t\}$ are the same in $I_1(t)$ and $I_2(j,t)$, let $\pi_{jk,t}$ denote the travel time realization of link (j,k) at time t in both I_1 and I_2. Following are notations used in the proof:

$$I_1' = A \times \{0, 1, ..., t, ..., t + \pi_{jk,t}\}, \quad \forall t \in T$$

$$I_2' = \delta(k) \times \{t + \pi_{jk,t}\}, \quad \forall k \in A(j), \forall t \in T$$

$$\bar{I}_1 = I_1' - I_1$$

$$I_{12} = \bar{I}_1 - I_2'$$

Please see Figure 2-3 for an intuitive representation of the relationship of these variables.

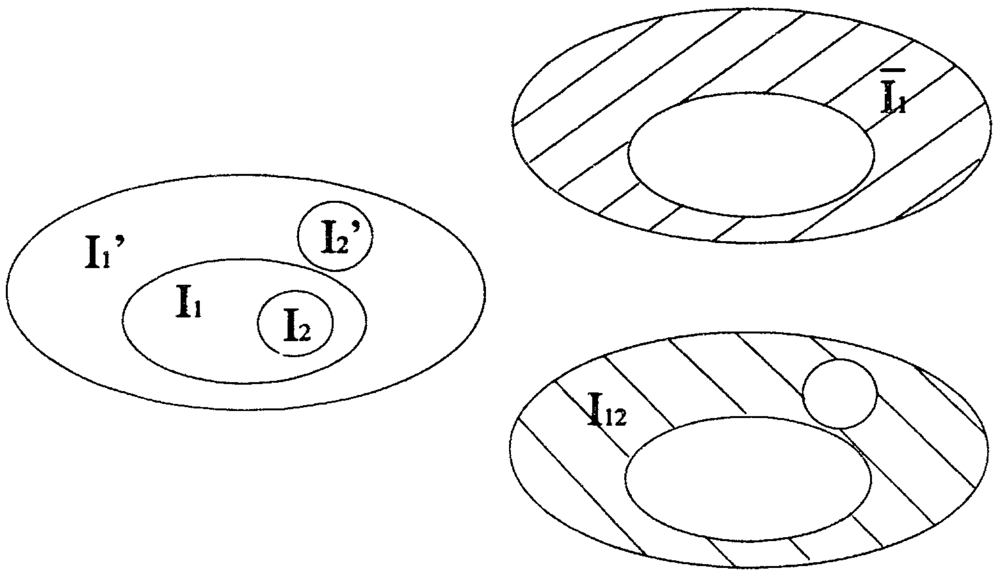


Figure 2-3: Relationship of $I_1, I_2, I_1', I_2', \bar{I}_1, I_{12}$

Induction Base: When $t \geq K - 1$, the network is deterministic and static. Therefore the BRP problem reduces to the classical shortest path problem where the shortest distance to the destination only depends on the origin node. Thus $e_{\mu_1^*}(j, t, I_1) = e_{\mu_2^*}(j, t, I_2), \forall I_1, I_2, \forall j \in N, \forall t \geq K - 1$.

Induction Assumption: Assume $e_{\mu_1^*}(j, t, I_1) = e_{\mu_2^*}(j, t, I_2), \forall I_1, I_2, \forall j \in N, \forall t \geq l$.

Induction Step: When $t = l - 1$,

$$
\begin{aligned}
& e_{\mu_1^*}(j, t, I_1) \\
= \; & \min_{k \in A(j)} \{ E_{C_{jk,t}|I_1} [C_{jk,t}|I_1 + E_{I_1'|I_1} [e_{\mu_1^*}(k, t + C_{jk,t}|I_1, I_1'|I_1)]]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_1'|I_1} [e_{\mu_1^*}(k, t + \pi_{jk,t}, I_1'|I_1)]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_1} [e_{\mu_1^*}(k, t + \pi_{jk,t}, I_1 + \bar{I}_1)]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_2'} [E_{I_{12}} [e_{\mu_1^*}(k, t + \pi_{jk,t}, I_1 + I_{12} + I_2')]]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_2'} [E_{I_{12}} [e_{\mu_2^*}(k, t + \pi_{jk,t}, I_2')]]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_2'} [e_{\mu_2^*}(j, t + \pi_{jk,t}, I_2')]\} \\
= \; & \min_{k \in A(j)} \{ \pi_{jk,t} + E_{I_2'|I_2} [e_{\mu_2^*}(j, t + \pi_{jk,t}, I_2'|I_2)]\} \\
= \; & \min_{k \in A(j)} \{ E_{C_{jk,t}|I_2} [C_{jk,t}|I_1 + E_{I_2'|I_2} [e_{\mu_2^*}(k, t + C_{jk,t}|I_2, I_2'|I_2)]]\} \\
= \; & e_{\mu_2^*}(j, t, I_2)
\end{aligned}
$$

The first equal sign is due to the definitions of $e_{\mu_1^*}$. The second equal sign is due to the definition of $\pi_{jk,t}$. The third equal sign is due to the statistical independence of link travel times in $\bar{I}_1$ and I_1. The fourth equal sign is due to the definition of I_{12} and the statistical independence of link travel times in I_{12} and I_2'. The fifth equal sign is due to the induction assumption. The sixth equal sign is due to the statistical independence of link travel times in I_{12} and I_2'. The seventh equal sign is due to the statistical independence of link travel times in I_2 and I_2'. The eighth equal sign is due to the definition of $\pi_{jk,t}$. The ninth (last) equal sign is due to the definition of $e_{\mu_2^*}$.

End of Proof

We can extend the theorem to the case when only part of the adjacent link travel time realizations are available. We conclude that current-information I for a given current-node and a given current-time in Group 1 is the **available** travel time realizations of **adjacent** links of the node at the current-time. Mathematically speaking, $I(j, t) = \delta(j) \cap IA$, where IA stands for information access, i.e. the available link travel time realizations. When the knowledge about the adjacent links of the current-node at the current-time is not available, the current-information becomes an empty set and the problem has the same current-information as that presented in the NI variant. Note that the name "NI" represents a variant where current-information I is an empty set. No information is only a sufficient condition to validate the specification of current-information. We choose "NI" as the name, as it is intuitive to get the idea of an empty current-information from the no-information assumption. However, we should remember that there are other conditions that can validate the "NI" formulation, one of which is discussed just now.

Variants with complicated information access and statistical dependency. Variants in Group 2 and Group 3 generally have complicated current-information. All available link travel time realizations are potentially useful and could be included in the current-information I. Network statistical dependency can be utilized to eliminate unnecessary link travel times from the current-information, as what we did in the independent case, but the judgment sometimes requires very smart work and the resulted dimension reduction of current-information may not compensate for the extra effort needed to distinguish them. In a word, the determination of current-information for variants in these two groups depends largely on the actual assumptions on both information access and network statistical dependency. In Section 2.4, we will discuss in more detail the perfect online information variants in Group 2.

Most transportation networks belong to Group 2 and Group 3. For example, a typical urban traffic network can be divided into several zones and we can assume that traffic within one zone is highly dependent, while weak relationship exists between traffic within the zone and that out of the zone. Furthermore, we can assume that only traffic conditions within the last one hour are helpful in predicting future. It is also very likely that there are several local traffic information centers that provide information to vehicles within their respective functional ranges. All these assumptions about network statistical dependency and information access complicated the problem, and careful problem definition is required.

We distinguish between Group 2 and Group 3, because the complexity of algorithms for variants in Group 2 and Group 3 could differ greatly. Complexity of an algorithm for the BRP problem depends largely on the maximum number of possible current-information realizations. For the sake of convenience of presentation, assume the partial on-line information is partial in the spatial dimension, not in the temporal dimension. With perfect on-line information, the current-information is composed of all link travel times up to current time t, and the maximum number of current-information realizations is just the maximum number of joint realizations of these tm random variables, which is at most R. With partial spatial on-line information, however, the current-information is composed of links around the path (what specific links are included depends on specific assumptions about "partial" spatial dependency) from the origin to the current-node. Therefore the current-information can take a maximum of $2^{tm} - 1$ different sets of links. As each set of link travel times has at most R joint realizations, the maximum number of current-information realizations is $(2^{tm} - 1)R$. The maximum numbers of current-information realizations in these two groups differ in a ratio of $2^{tm} - 1$, which is significant. It is stated in [25] that in a static network, the maximum number of current-information realizations with partial on-line information is $2^R - 1$. This is a quite loose upper bound, and a tighter upper bound obtained by applying the above logic would be $(2^m - 1)R$.

The dynamic shortest path problem in acyclic networks with independent stationary Markovian arc costs studied in [26] can be viewed as an infinite horizon version of a variant in Group 3. Please refer to the literature review of Section 2.1 for an introduction of the basic assumptions. The assumption of acyclic networks implies that node j cannot be visited again after the traveler leaves it. Since the Markovian arc costs are independent across nodes, it is not helpful to keep information of any already visited nodes. Thus the dimension problem of current-information with partial spatial online information as discussed above does not exist in this case. This assumption along with the stationary assumption makes a polynomial running time algorithm possible.

2.3 The No-Information Variant

In this section, we discuss in details the no-information (NI) variant of the best routing policy problems in STD networks. In subsection 2.2.2, we defined the no-information variant as a variant of the BRP problem where the current-information component of any state is an empty set. We also discussed several situations under which the NI variant is applicable. These situations include:

- When no knowledge about any of the link travel time realizations is available
- When all link travel time random variables are statistically independent *and* no knowledge about the realizations of outgoing link travel times of current node at current time is available

We call this definition of NI variant the **special definition**. A direct implication of this definition is that current-information will not appear in the optimality conditions. In other words, routing decisions only depend on the current node and the current time. We deem that the decision dependency (i.e. what the routing policies are based on) is the key in defining a variant, as it directly affects the algorithm design. In light of this, a general definition of NI variant would be: the current-information component of any state is the same. In this case, routing decisions also only depend on the current node and the current time. In the rest of the thesis, the NI variant is defined with the special definition, otherwise indicated.

2.3.1 Motivation

There are three kinds of motivation for studying the no-information variant. Theoretically, the NI variant is the simplest in terms of algorithm design among all BRP problem variants with on-line information, due to the lack of current-information. It is therefore the basis for the study of more complicated variants. Furthermore, even though it is the simplest, it suffices to show some of the implications and significance of stochasticity in a dynamic context for traffic models. It also shows the fact that how information access can affect the routing problem formulation. Practically, there does exist quite a few traffic situations where the NI formulation is applicable. For example, in a network where link travel times are weakly coupled, or where little or no information is available. Computationally, the NI variant can be solved in polynomial time, as shown later in the section. This is a very desirable result, as the BRP problem in a STD network generally requires exponential running time to solve. Therefore NI can be used as an approximation to more complicated variants, and we will discuss this in great details in Chapter 3.

2.3.2 Optimality Conditions

The optimality conditions have been presented in subsection 2.2.2. We list them here for the convenience of reference:

$$e_{\mu^*}(j,t) = \min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.6}$$

$$\mu^*(j,t) = \arg \min_{k \in A(j)} \{E_{C_{jk,t}}[C_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.7}$$

with the boundary conditions: $e_{\mu^*}(d,t) = 0, \forall t \in T$, and $e_{\mu^*}(j,t) = e_{\mu^*}(j, K-1), \forall j \in N, \forall t > K-1$.

We can image the traveler in a network whose level of uncertainty never decreases. The traveler's knowledge about the network remains as the *a prior* distribution of link travel times, either because he/she has no en route information access, or because the network is statistically independent and online information cannot help to predict the future. Thereafter, one can work only with the unconditional marginal distributions of link travel times, as shown in the optimality conditions, either because there is nothing to be conditional on, or because the conditional probabilities are the same as the unconditional probabilities.

We will work on an illustrative example to show how NI optimality conditions work. Please see Figure 2-4 for the network and link travel time data. The topological network is shown at the upper-left corner of the figure, and the major part of the figure is a time-space representation of the network. In a time-space network, time is marked along the vertical axis (the so-called time axis), and node number is marked along the horizontal axis (the so-calld space axis). Each point in this network represents a node-time pair (j, t), and any link between (j, t_1) and (k, t_2) indicates that link (j, k) has a travel time of $t_2 - t_1$ if departure time from node j is t_1. We are interested in finding the best routing policy from node 1 to node 4 at departure time 0, namely $e_{\mu^*}(1, 0)$, and only those node-time pairs and links relevant to the computation are shown.

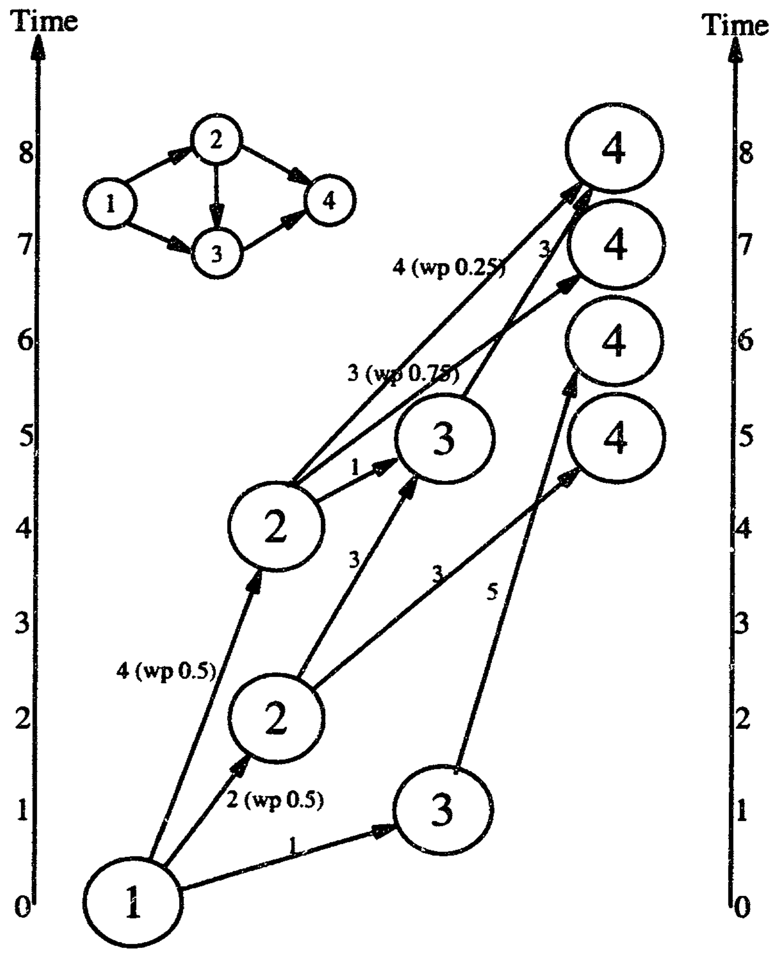


Figure 2-4: An Illustrative Example for NI Optimality Conditions: Topological Network and Time-Space Network

Figure 2-4 shows the marginal distribution of link travel time random variables. Link $(1, 2)$ at time 0 could have two realizations of travel time: 4 *w.p.* 0.5 and 2 *w.p.*

0.5. Link $(2,4)$ at time 4 could have two realizations of travel times: 4 *w.p.* 0.25 and 3 *w.p.* 0.75. All other link travel times are deterministic.

We apply the optimality conditions to obtain the value of $e(1,0)$.

$$e_{\mu^*}(1,0) = \min\{1 + e_{\mu^*}(3,1), 0.5 \times (2 + e_{\mu^*}(2,2)) + 0.5 \times (4 + e_{\mu^*}(2,4))\}.$$

It can be easily observed from the figure that $e_{\mu^*}(3,1) = 5$ and $\mu^*(3,1) =$ (node) 4, and $e_{\mu^*}(2,2) = 3$ and $\mu^*(2,2) =$ (node) 4. We apply the optimality condition again to obtain $e_{\mu^*}(2,4)$:

$$e_{\mu^*}(2,4) = \min\{1 + e_{\mu^*}(3,5), 0.25 \times 4 + 0.75 \times 3\} = \min\{1 + 3, 1 + 2.25\} = 3.25$$

and $\mu^*(2,4) =$ (node)4. With the values of $e_{\mu^*}(3,1), e_{\mu^*}(2,2)$, and $e_{\mu^*}(2,4)$ in hand, we can obtain

$$e_{\mu^*}(1,0) = \min\{1 + 5, 0.5 \times (2 + 3) + 0.5 \times (4 + 3.25)\} = 6$$

and $\mu^*(1,0) =$ (node) 3. Therefore the optimal routing policy for node 1 at time 0 turns out to be a path: 1-3-4.

2.3.3 Algorithm DOT-S and Algorithm LC

We can associate with each pair (j,t) a label which is the upper bound of the minimum expected travel time from node j to the destination node d at departure time t. We will design a procedure to update these labels according to the optimal conditions, until all of them are optimal. Depending on the way the labels are updated, there are two different algorithms.

Algorithm DOT-S is a counterpart of Algorithm DOT [9] which finds the shortest path in a deterministic time-dependent network. DOT stands for "Decreasing Order of Time", and S stands for "Stochastic". It is noted that the update of labels at time t depends only on labels at times later than t, due to the assumption of positive link travel times. Therefore, we can first solve a classical shortest path problem for the deterministic and static period where link travel times are $c_{jk,K-1}, \forall (j,k) \in A$, and set $e_{\mu^*}(j, K-1)$ = shortest path length from j in the classical SSP problem, $\forall j \in N$. We then proceed to the labels of time $K-2$ which only depend on labels of time $K-1$. As labels of time $K-1$ is already optimal, by optimality condition 2.6, the updated labels of time $K-2$ are also optimal. We continue this procedure back in time until time 0, and every label will then be set to be optimal.

We define $\tau^v_{jk,t}$ as the v^{th} realization of the marginal distribution of travel time of link (j,k) at time t, and $q^v_{jk,t}$ the corresponding marginal probability. We also define Q as the maximum number of realizations for a single link travel time marginal distribution. The statement of Algorithm DOT-S is as follows:

Algorithm DOT-S

```
Step 0: (Initialization)
        0.1: Run the classical shortest path problem algorithm (e.g. Dijkstra's)
             on the deterministic and static network $G'(N, A)$
             where link $(j,k)$ has a travel time of $c_{jk,K-1}, \forall (j,k) \in A$;
        0.2: $e_{\mu^*}(j, K-1)$ = Shortest path length from node $j$ to node $d$;
        0.3: $e_{\mu^*}(j,t) = \infty, \mu^*(j,t) = \infty, \forall j \in A - \{d\}, \forall t < K-1$;
             $e_{\mu^*}(d,t) = 0, \forall t \in T$.
Step 1: (Main loop)
        for $t = K-1$ to 0
             for $(j,k) \in A$
                  $temp = \sum_v (\tau^v_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})) \times q^v_{jk,t}$;
                  If $temp < e_{\mu^*}(j,t)$
                       $e_{\mu^*}(j,t) = temp$
                       $\mu^*(j,t) = k$
```

Let us now make a comparison between solutions from Algorithm DOT-S and Algorithm DOT. They look similar, as each pair (j,t) has an associated cost to the destination node, and an associated next node to take. The difference can be obtained by tracing a traveler in the network. Assume a traveler starts from the pair (j,t) in a deterministic time-dependent network and follows the optimal routing decisions computed from Algorithm DOT. As the travel times are deterministic, we can tell for sure when he/she will arrive at downstream nodes and thus the path he/she will take can be determined. Instead of telling him/her to make routing decisions based on current node j and current time t, one can just tell him/her to follow an *a priori* path. However, if the traveler travels in a STD network with no information access, one cannot tell what path he/she will end up following before the trip begins, as the link travel times are random. To put in other words, the traveler could arrive at downstream nodes at several possible times. Therefore the traveler must have the routing policy $\mu^*(j,t)$ computed from Algorithm DOT-S and make decisions depending on arrival times.

The complexity analysis of Algorithm DOT-S is straightforward. At the initialization period, a classical shortest path algorithm is run and the running time is $\theta(SSP)$, where SSP is the running time of a classical shortest path algorithm (cf [1] for a summary of running times of difference algorithms). In the main loop, at each time period of the dynamic period (i.e. $t < K-1$), each arc is visited exactly once with Q arithmetic operations, and each node is visited at least once and at most three times. Therefore the running time of the main loop is $\theta(nK + mQK)$. To sum up, the complexity of Algorithm DOT-S is $\theta(SSP + nK + mQK)$.

There are other kinds of algorithms that implement the optimality conditions to solve the NI variant. One of them is a straightforward extension of the label correcting algorithm for the classical shortest path problem. Miller-Hooks and Mahmassani [20] presented such an algorithm. The statement of the algorithm can be rewritten in our notation as follows. We will denote it as Algorithm LC.

Algorithm LC

```
Step 0: (Initialization)
        0.1: Initialize node labels
             $e_{\mu^*}(j,t) = \infty, \mu^*(j,t) = \infty, \forall j \in A - \{d\}, \forall t < K-1;$
```

```
            $e_{\mu^*}(d,t) = 0, \forall t \in T.$
        0.2: *Initialize the scan-eligible list*
            Create the scan-eligible list SE, and insert node $d$.
**Step 1:** (Choose Current Node)
        If the SE list is empty, stop.
        Otherwise, select the first node from the SE list.
        Call this node the current node $k$.
**Step 2:** (Update the Node Labels)
        For each $j \in B(k)$
            For each $t \in T$
                $temp = \sum_v (\tau^v_{jk,t} + e_{\mu^*}(k, t + C_{jk,t})) \times q^v_{jk,t}$;
                If $temp < e_{\mu^*}(j,t)$
                    $e_{\mu^*}(j,t) = temp$
                    $\mu^*(j,t) = k$
                    If $j \notin$ SE, put $j$ in SE list.
```

According to Miller-Hooks and Mahmassani [20], the LC algorithm with a basic FIFO SE list has a worst-case computational complexity of $O(K^2n^3Q)$. The readers are referred to their paper for a detailed proof of the result.

2.3.4 Extension to Minimum Expected Cost Problems

We have so far focused on the minimum expected travel *time* problem. In fact, the minimum expected travel cost problem can be handled with straightforward extension. Define a link cost function $g(C_{jk,t})$ to be the cost of link (j,k) at time t as a function of link travel time $C_{jk,t}$, and particularly $g(0) = 0$. The minimum expected cost problem is to find a routing policy with minimum expected cost from all origins for all departure times to the destination node.

The optimality conditions for the minimum expected cost problem can be obtained by making slight changes from those for the minimum expected travel time problem.

For the sake of notation simplification, we will still use $e_\mu(j,t)$ to denote the expected cost of a routing policy μ with origin node j, departure time t, and empty current-information. The optimal routing policy μ^* and the corresponding optimal expected cost e_{μ^*} are solutions of the following system of equations:

$$e_{\mu^*}(j,t) = \min_{k \in A(j)} \{E_{C_{jk,t}}[g(C_{jk,t}) + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.8}$$

$$\mu^*(j,t) = \arg\min_{k \in A(j)} \{E_{C_{jk,t}}[g(C_{jk,t}) + e_{\mu^*}(k, t + C_{jk,t})]\} \tag{2.9}$$

with the boundary conditions: $e_{\mu^*}(d,t) = 0, \forall t \in T$, and $e_{\mu^*}(j,t) = e_{\mu^*}(j, K-1), \forall j \in N, \forall t > K-1$.

Algorithms for the minimum expected cost problem can be obtained similarly. We can see that algorithms for the "cost" problem has the same asymptotic running times as those for the "time" problem, as the only additional operation of the "cost" problem is the mapping from $C_{jk,t}$ to $g(C_{jk,t})$. In actual implementation, the mapping can be done in the data generation. For example, for each realization of $C_{jk,t}$, one can generate a cost realization associated with it. In this case, the "cost" problem algorithms and the "time" problem algorithms have exactly the same running times.

2.3.5 Comparison of Running Times of Algorithm DOT-S and LC

Algorithm DOT-S and Algorithm LC are two different ways to apply the same optimality conditions. They differ in the order the node labels are checked. Algorithm DOT-S checks labels in decreasing order of time, while Algorithm LC uses a scan eligible list to maintain active nodes and the checking order is primarily topological. Using the terminology of network optimization [1], Algorithm DOT-S is a label-setting algorithm, while Algorithm LC is a label-correcting algorithm. A label-setting algorithm sets a label to its optimal value at the first time the label is updated, while a label-correcting algorithm needs to run several passes over the labels to have them optimal. The reason that Algorithm DOT-S can be a label-setting algorithm is the assumption of positive link travel times, as discussed before.

It is shown by Chabini [10] that Algorithm DOT-S is optimal, in the sense that no other algorithm can have better theoretical complexity. The argument is as follows. In order to make sure the solution is optimal, any algorithm has to retrieve the data at the dynamic period, i.e. $t < K - 1$, for at least once. The data for the problem is the discretized marginal probability distributions for all links at all times lower than $K - 1$. Thus the retrieve of data takes a running time of $\theta(mKQ)$. Furthermore, any solution algorithm must in the worst case compute and output, or in the very least initialize $\theta(nK)$ variables consisting of the values of $e_{\mu^*}(j, t)$ and $\mu^*(j, t)$ for all pair (j, t). Finally, computing all-to-one least expected travel times for departure times beyond the time horizon $K - 1$, is equivalent to computing an all-to-one shortest path tree using $c_{jk,K-1}$ as link travel times. In summary, any solution algorithm to the NI variant has a worst-case complexity of $\Omega(SSP + nK + mKQ)$, which is the same as the worst-case complexity of Algorithm DOT-S. Following the claim, we can conclude that *asymptotically* Algorithm DOT-S has a running time at least as good as Algorithm LC does.

Extensive computational tests have been carried out. The objectives of the computational tests are: to study experimentally the running times of the two algorithms as functions of various network parameters, and to compare the actual running times of Algorithm DOT-S and Algorithm LC. In the rest of the subsection, detailed description of the computational tests will be provided.

Random Network Generator

The random network generator generates a random directed network on which the algorithms are to be applied. Two sets of data have to be generated, the topology of the network and the discretized link travel time distributions. To generate the topology of the network, the required input from the users is: 1) the number of nodes n; 2) the number of links m; 3) the maximum in-degree; and 4) the maximum out-degree. By default, the node with the highest number is set to be the destination node. To assure connectivity to the destination node, a directed in-tree rooted at the destination node is generated at the first place. The remaining $m - (n - 1)$ links are generated by selecting the head node and tail node randomly, assuring that the maximum in-degree and out-degree constraints are satisfied.

To generate the discretized link travel time distributions, the required input from the users is: 1) the number of time periods K; 2) the number of realizations for a single link at a given time Q; 3) the maximum of link travel time realizations; 4) the minimum of link travel time realizations; 5) the maximum of link cost realizations; and 6) the minimum of link cost realizations. For each time point and each link, two sets of numbers are generated, the first set contains Q random numbers in the range of the given minimum and maximum link travel time realizations, and the second set contains Q random numbers in the range of 0 and 1. The first set are the link travel time realizations for the specific link at the specific time, and the second set normalized by the sum of the Q numbers are the marginal probabilities associated with each realization.

Tests Design

Basically the tests can be divided into two parts: those on sparse networks which are usually the cases for transportation networks, and those on dense networks. For the tests on sparse networks, we set the ratio of the number of links to the number of nodes to a constant of 3. The maximum link travel time is 25, and the minimum link travel time is 1. The maximum link travel cost is 40, and the minimum travel cost is 1. We examine three different topologies of networks: 100 nodes, 500 nodes, and 1000 nodes. For each topology, there are 3 different numbers of realizations (R): 5, 10, and 20 and 3 different numbers of time periods (K): 30, 60, and 90. Therefore there are 9 different sets of link travel time/cost data for a given topology, and altogether there are 27 experiments for sparse networks. We define an experiment as a series of runs with the same topological and link travel time/cost data, namely with the same triple (n, Q, K). 10 independent runs are carried out for each of the 27 experiment. For example, for the experiment with the triple $(n, Q, K) = (100, 5, 30)$, 10 different random networks are generated with the destination node fixed as the last node. In each run, the running time in CPU seconds for both Algorithm LC and Algorithm DOT-S **for the minimum expected cost problem** are recorded, and their ratio recorded, too. We then take average of the running times and their ratio over the 10 runs.

For the tests on dense networks, we fix the number of nodes to be 100, and have three different values for the number of links: 1000, 2500, and 5000, with average in- and out-degree of 10, 25, and 50 respectively. The maximum link travel time and maximum link cost are both $2 \times Q$, and the minimum link travel time and minimum link cost are both 1. We will discuss later in the tests results why we choose $2 \times Q$ rather than a fixed number. There are 3 different numbers of realizations that are the same as those in the sparse network tests: 5, 10, and 20. The values that the number of time periods can take are different from those in sparse tests: 60, 120, and 240. Therefore in the dense tests, an experiment is defined by a different triple (m, Q, K). Similarly, 10 independent runs are carried out for each experiment and averages of running times for both algorithms and their ratios are taken.

Algorithm DOT-S and Algorithm LC are implemented in C++ and complied by GNU C++ complier. We use the classical label correcting algorithm with a complexity of $O(nm)$ to compute the static shortest path at the static period for Algorithm DOT-S. The codes are run on a Dell OptiPlex GX100 workstation with 933MHz CPU, 256 megabytes RAM, running Red Hat Linux 7.0 operating system.

Tests Results for Sparse Networks

The test results for sparse networks are shown in Table 2.3 and in Figure 2-5 through Figure 2-10. First of all, we study respectively the running times of Algorithm DOT-S and Algorithm LC as functions of network parameters: m, K, and Q. We first discuss Algorithm DOT-S, and then Algorithm LC. At last we compare the two algorithms: how their relative running time varies with network parameters.

We only present some typical results in figures, as other results have similar features. Figure 2-5 show the running time of Algorithm DOT-S as a function of the number of links (m), with the number of time periods (K) fixed at 60, and for all

DOT-S

# nodes	# links	# realizations	$K = 30$	$K = 60$	$K = 90$
100	300	5	0.012	0.026	0.041
		10	0.025	0.049	0.070
		20	0.045	0.089	0.133
500	1500	5	0.093	0.194	0.296
		10	0.146	0.305	0.463
		20	0.251	0.519	0.780
1000	3000	5	0.204	0.419	0.637
		10	0.308	0.628	0.966
		20	0.518	1.050	1.601

Label Correcting

# nodes	# links	# realizations	$K = 30$	$K = 60$	$K = 90$
100	300	5	0.021	0.045	0.070
		10	0.029	0.056	0.101
		20	0.050	0.101	0.150
500	1500	5	0.108	0.231	0.380
		10	0.151	0.305	0.476
		20	0.256	0.510	0.753
1000	3000	5	0.222	0.485	0.778
		10	0.308	0.673	0.992
		20	0.513	1.033	1.570

Ratio(LC/DOT-S)

# nodes	# links	# realizations	$K = 30$	$K = 60$	$K = 90$
100	300	5	1.90	1.82	1.72
		10	1.23	1.15	1.44
		20	1.13	1.14	1.13
500	1500	5	1.16	1.19	1.28
		10	1.03	1.00	1.03
		20	1.02	0.98	0.97
1000	3000	5	1.09	1.16	1.22
		10	1.00	1.07	1.03
		20	0.99	0.98	0.98

Table 2.3: DOT-S vs. LC: Summary of Running Times (CPU sec.) – Sparse Networks (#links/#nodes = 3)

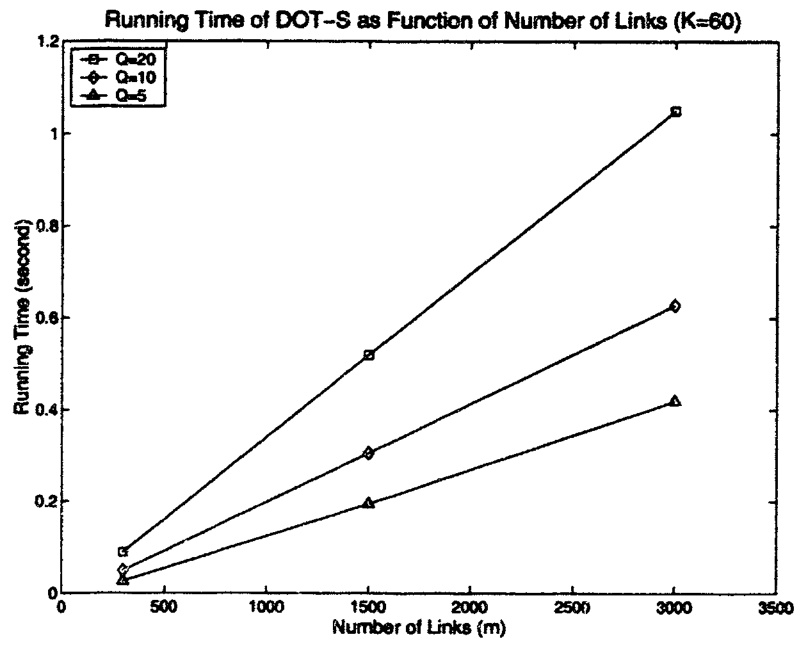


Figure 2-5: Running Time of DOT-S as Function of Number of Links (K=60)

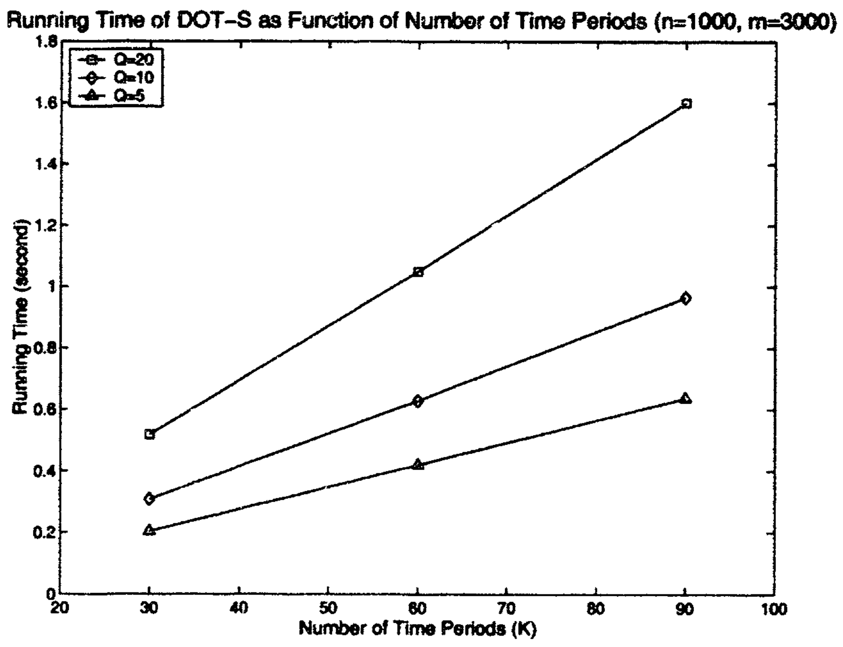


Figure 2-6: Running Time of DOT-S as Function of Number of Time Periods (n=1000, m=3000)

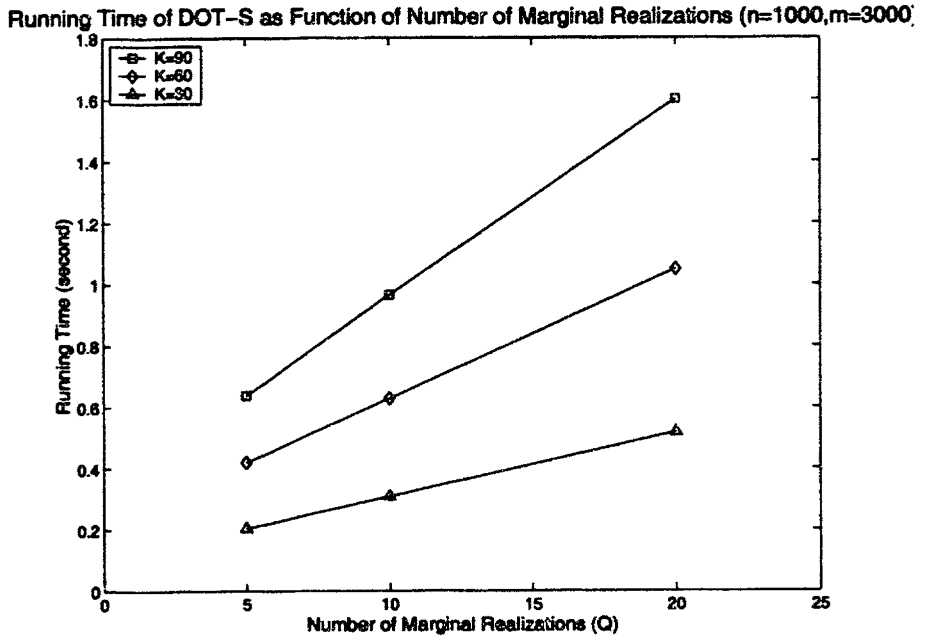


Figure 2-7: Running Time of DOT-S as Function of Number of Realizations (n=1000, m=3000)

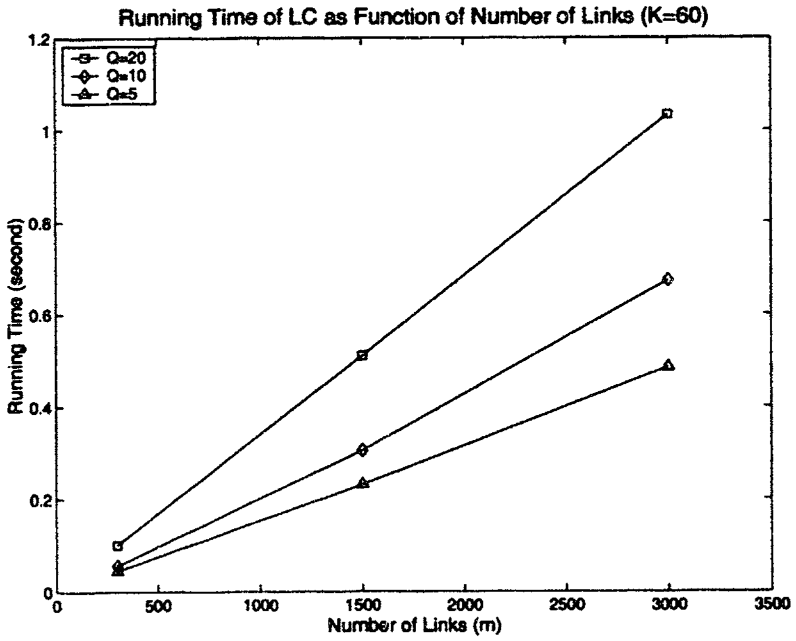


Figure 2-8: Running Time of LC as Function of Number of Links (K=60)

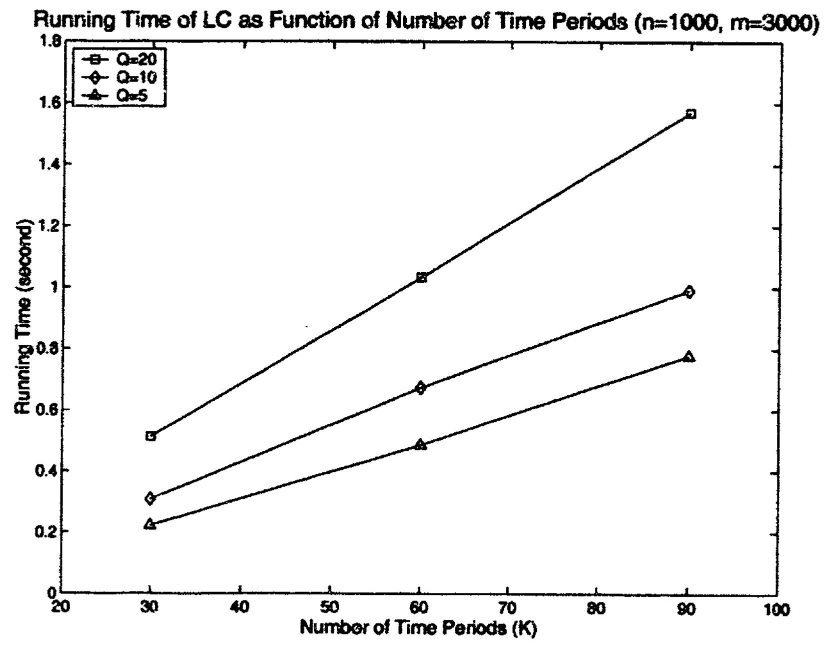


Figure 2-9: Running Time of LC as Function of Number of Time Periods (n=1000, m=3000)

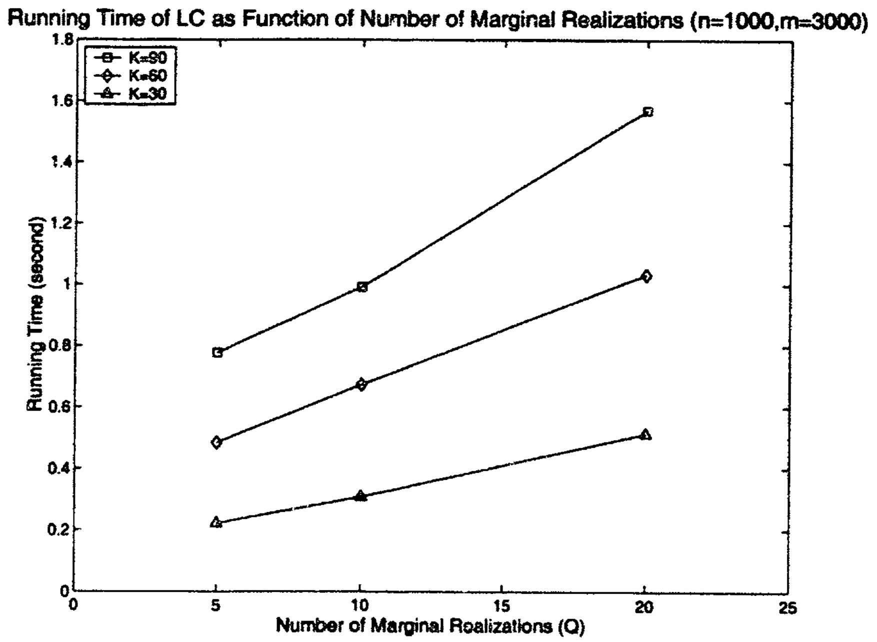


Figure 2-10: Running Time of LC as Function of Number of Realizations (n=1000, m=3000)

three possible number of marginal realizations: 20, 10, and 5. We can see that the running time of Algorithm DOT-S increases linearly with the number of links for all three Q values. We can also see this nearly perfect linear relationship with respect to the number of time periods and the number of realizations in Figure 2-6 and Figure 2-7 respectively. These results are consistent with the theoretical analysis which gives a running time of $\theta(SSP + nK + mKQ)$. As Algorithm DOT-S is a dynamic-programming-type algorithm, the actual running time can be accurately analyzed. This explains the closeness between the theoretical and experimental results.

Figure 2-8, Figure 2-9 and Figure 2-10 show the running times of Algorithm LC as functions of the number of lin_s (m), the number of time periods (K), and the number of realizations (Q) respectively. The relationship is roughly linear, which is much better than the worst-case complexity predicted by the theoretical analysis. This is not quite surprising, because label correcting algorithms usually have different actual running times depending on network topology, cost structure and data structure implementation. If we look at the analysis of Algorithm LC [20] in details, we find that the worst case happens if: 1) only one label is permanently set during one pass over all nodes. As there are $(n-1)K$ labels to be set and each pass contains $(n-1)$ updates, there are altogether $(n-1)^2K$ updates; 2) the network is completely connected, i.e. each node is connected to every other links, so that each update requires $(n-1)KQ$ computations. In actual cases, it is very likely that more than one label will be permanently set during one pass, and that each pass contains less than $(n-1)$ updates and that the network is far from being completely connected. Indeed, the optimal next arc choice for a given node at a given time points is usually also optimal for other time points for the same node, so possibly $\theta(K)$ labels can be permanently set during one pass. Furthermore, in a sparse network such as the one used in the tests, the average degree is $\theta(1)$ rather than $\theta(n)$. Please note in the sparse network tests, we fix the ratio of number of links to number of nodes, so in the theoretical analysis $\theta(m) = \theta(n)$.

Next we study the relative running times of the two algorithms. As we discussed before, Algorithm DOT-S has an optimal theoretical running time among all solution algorithms for the NI variant. In the sparse network tests, we can see in Table 2.3 that Algorithm DOT-S is more efficient than Algorithm LC for most of the experiments. In fact, every operation of Algorithm DOT-S in the dynamic phase (i.e. $t < K - 1$) must be performed in Algorithm LC, and the initialization of the two algorithms are the same. The only difference lies in the way the labels in the static period is computed. Specifically in our implementation, Algorithm DOT-S uses a classical label-correcting algorithm with FIFO scan eligible list to compute the static shortest path, while Algorithm LC has the computations for the static period and dynamic period bundled together. Although the static shortest path problem is solved by label-correcting algorithm in both algorithms, the actual orders in which the nodes are scanned and labels are updated can be different, therefore the relative running times can be either greater or less than 1. To sum up, in the dynamic phase, Algorithm DOT-S needs no more running time than Algorithm LC does, and in the static phase, either algorithm possibly needs more running time than the other. This explains the phenomena that for some experiments, the running time ratio of LC to DOT-S is less than 1, e.g. when $n = 1000, m = 3000, Q = 20$. If traveling beyond the dynamic phase is prohibited, Algorithm DOT-S always performs at least as well as Algorithm LC does.

Now we will see how the ratio varies with network parameters: K, Q, and $m(n)$. The order in which the labels are set in Algorithm DOT-S is optimal, while Algorithm LC does some overhead work in label updating. Therefore the ratio of LC/DOT-S represents to some extent the overhead work Algorithm LC does in additional to the necessary work. Our discussion below can also be viewed as about the efficiency of Algorithm LC with Algorithm DOT-S as the benchmark. Here we will provide some intuition on the label correcting algorithm. They are not aimed not be rigorous, though. Rather they are to be helpful in understanding the behavior of Algorithm LC in a high level. As we use a FIFO queue to implement the scan eligible list, we are actually scanning the nodes roughly in the order of breadth first search (BFS). It is well known that a breadth first search tree is a shortest path tree if the link travel costs are the same for all links in a static and deterministic network. In a dynamic and stochastic network, this argument requires more discernment. However, a general idea is that more evenly distributed link travel cost data will very likely yield less overhead label updating work. On the other hand, if the network itself is a tree, no matter what the link travel costs are, the shortest path tree will be the tree itself. Therefore the topology of the network also matters a lot in label correcting algorithms. Specifically the average degree determines to some extent how far away a final shortest path tree can deviate from a breadth first search tree. This effect will be discussed in details in the dense network tests.

There is no definite relationship between the ratio of running times of Algorithm LC and Algorithm DOT-S with respect to the number of time periods, K, as shown in Table 2.3. This is intuitively correct, since we can view computations of additional time periods as replications of those for earlier periods. This is not to say that they are identical. Rather it says that no fundamental changes in data structure or topology exist if we change the number of time periods from 30 to 60.

Next we can see that the ratio of running times of Algorithm LC and Algorithm DOT-S decreases with respect to the number of realizations for a single link travel cost marginal distribution, Q. Note that link costs have uniform distributions within a fixed range, in the case, from 1 to 40, so the variance of the continuous uniform distribution is fixed. We are actually sampling from this distribution and Q is the sample size. It is well known that the variance of sample mean is inversely related to the sample size and accordingly the standard deviation of sample mean is inversely related to the square root of sample size. Therefore with larger Q, the link travel costs are more evenly distributed and the label updating overhead is less.

Finally the ratio of running times of Algorithm LC and Algorithm DOT-S decreases with the number of links, m. When networks are scaled up with the same link cost structure and topology, the average number of links in a routing policy is larger and the average expected travel cost of a routing policy is larger. Therefore intuitively the travel cost of a path (policy) is more dependent on the number of links in that path (policy) and thus a BFS is a more optimal order of scanning the nodes.

Tests Results for Dense Networks

The test results for dense networks are shown in Table 2.4 and in Figure 2-11 through Figure 2-14. First of all, we study the running times of Algorithm DOT-S and Algorithm LC as functions of network parameters d which is defined as the average degree. Next we compare the two algorithms: how their relative running time varies with network parameters.

Figure 2-11 through Figure 2-14 show the running times of Algorithm DOT-S and Algorithm LC as functions of average degree. As the number of nodes is fixed, this relationship is actually with respect to the number of links m. For algorithm DOT-S, this relationship should be linear asymptotically, yet we observe a relationship a little bit worse than linear in Figure 2-11 and Figure 2-12. This is due to the overhead of shortest path computation in the static phase. Note we use a label correcting algorithm in the static phase, and the actual running time of a label correcting algorithm increases more than linearly with average degree. In Figure 2-13 and Figure 2-14, we see a relationship even worse than linear, and this is consistent with the general conception than the actual running time of label-correcting-type algorithm is greatly affected by the average degree of a network. Indeed, if each node has more incoming arcs, the update of its label will potentially affect more nodes and therefore potentially more nodes will enter the scan eligible list more than once. However, this effect is constrained by the variability of link costs. At the extreme case, when all links have the same cost, the average degree does not affect the actual running time at all.

Next let us look at the ratio of running times of Algorithm DOT-S and Algorithm LC. Note that in the dense network tests, the range of the uniform distribution for link travel costs are proportional to the number of realizations Q. This is due to the fact that with fixed uniform distribution range, the actual running of Algorithm LC for $Q = 10$ is even less than that for $Q = 5$. This shows that the sample size effect is rather significant. In order to counter this effect to some extent, we set this new range. As shown from the ratio table, the difference with respect to Q is less dramatic than that shown in Table 2.3 where the uniform distribution range is fixed (compare the case for $K = 60$ and $n = 100$).

The relationship between the ratio and average degree is not obvious. Indeed two

DOT-S

# nodes	# links	# realizations	$K = 60$	$K = 120$	$K = 240$
100	1000	5	0.097	0.276	0.551
		10	0.183	0.439	0.903
		20	0.324	0.786	1.635
	2500	5	0.194	0.564	0.140
		10	0.372	0.921	1.842
		20	0.668	1.633	3.239
	5000	5	0.412	1.188	2.364
		10	0.756	1.904	3.727
		20	1.347	3.302	n/a

Label Correcting

# nodes	# links	# realizations	$K = 60$	$K = 120$	$K = 240$
100	1000	5	0.173	0.459	1.092
		10	0.267	0.524	1.392
		20	0.445	0.916	2.201
	2500	5	0.379	0.986	2.218
		10	0.527	1.085	2.945
		20	0.941	1.820	4.404
	5000	5	0.800	2.044	4.712
		10	1.076	2.413	6.496
		20	1.876	3.776	n/a

Ratio(LC/DOT-S)

# nodes	# links	# realizations	$K = 60$	$K = 120$	$K = 240$
100	1000	5	1.79	1.66	1.98
		10	1.46	1.19	1.54
		20	1.37	1.16	1.35
	2500	5	1.96	1.75	1.95
		10	1.42	1.18	1.60
		20	1.41	1.12	1.36
	5000	5	1.94	1.72	1.99
		10	1.42	1.27	1.74
		20	1.39	1.14	n/a

Table 2.4: DOT-S vs. LC: Summary of Running Times (CPU sec.) – Dense Networks(#links/#nodes $\geq$ 10)

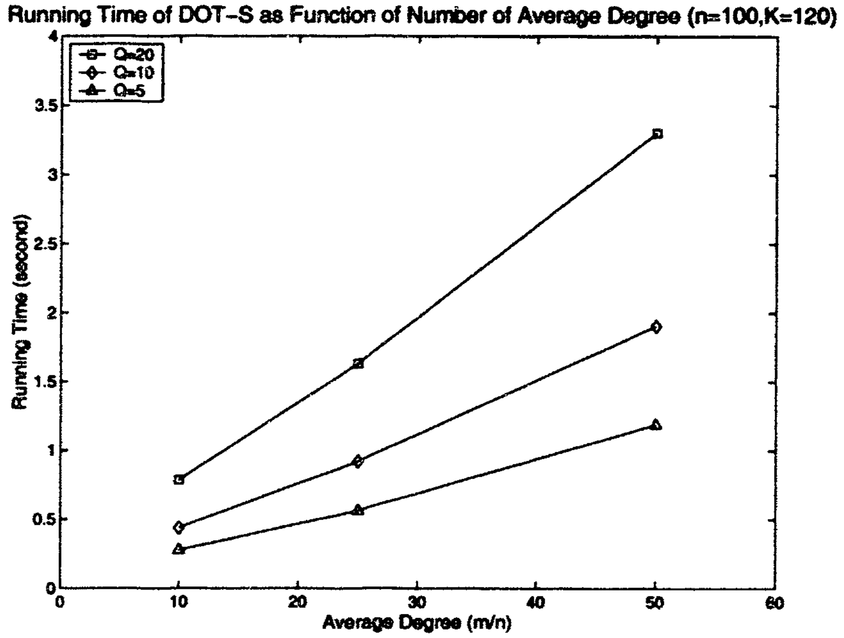


Figure 2-11: Running Time of DOT-S as Function of Average Degree (n=100, K=120)

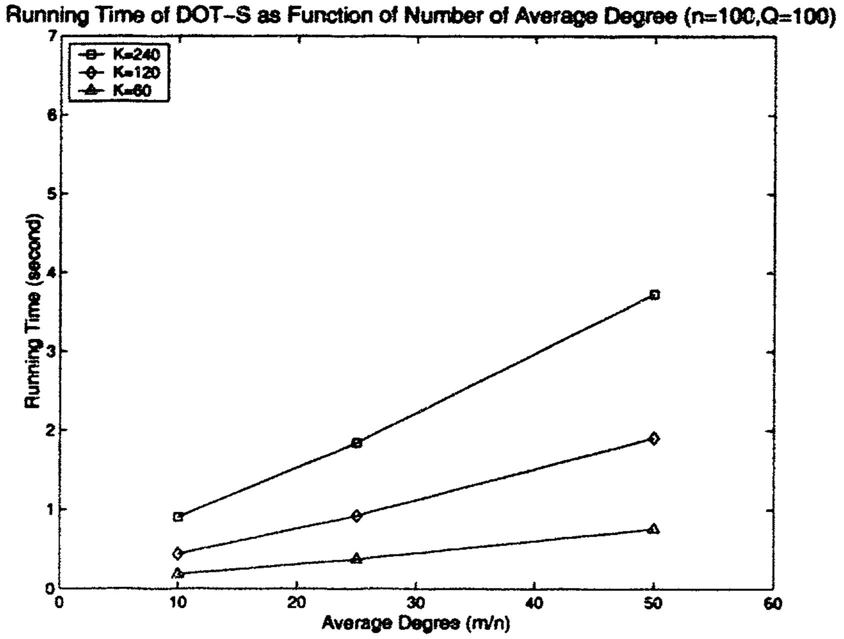


Figure 2-12: Running Time of DOT-S as Function of Average Degree (n=100, Q=10)

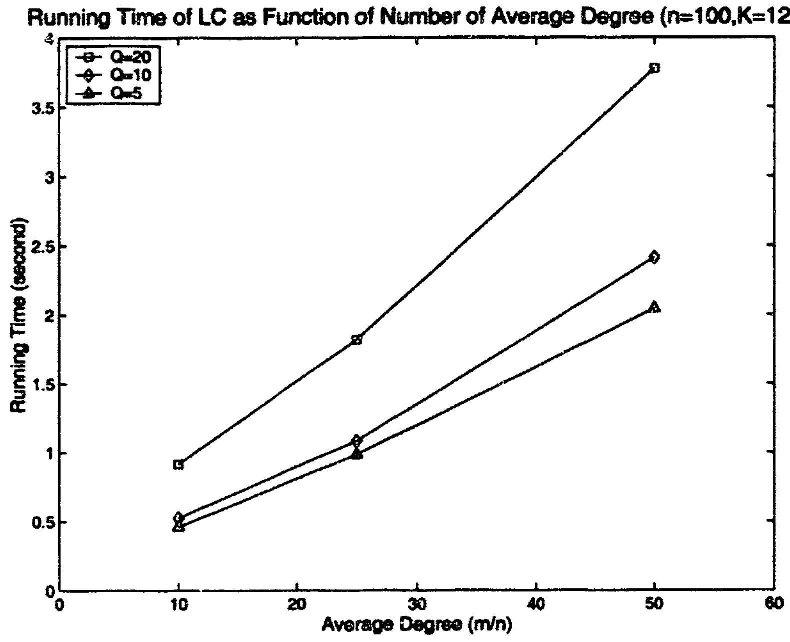


Figure 2-13: Running Time of LC as Function of Average Degree (n=100, K=120)

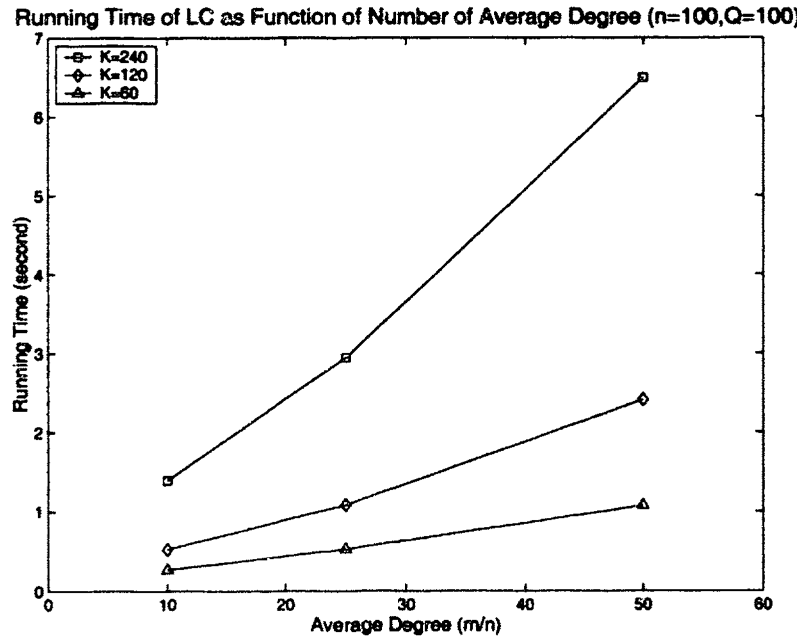


Figure 2-14: Running Time of LC as Function of Average Degree (n=100, Q=10)

forces are countering each other here. As discussed before, larger average degree will bring more overhead in label updating, while more links will smooth out the difference in link costs and make the cost of a path (policy) more dependent on the number of links on it. With larger average degree and therefore larger number of links in this case, the first force makes Algorithm LC more time-consuming while the second force makes it less time-consuming. Further test where the number of links are fixed with varying average degrees can potentially show the ratio as a function of average degree.

We note that the ratio is greater than 1 (i.e. Algorithm DOT-S outperforms Algorithm LC) for all dense network tests. This is because the effect of static phase shortest path computation is less significant here. However, one of the conclusions in [21] is that in dense networks, Algorithm LC outperforms Algorithm DOT-S. Further details of computation tests in [21] is needed to determine the cause of this counter-theory results.

2.4 The Perfect Online Information Variant

In the previous section, we studied the no-information (NI) variant in details. The assumption of empty current-information is not so realistic in the presence of Advanced Traveler Information System (ATIS) and/or Advanced Traffic Management System (ATMS). On the other hand, a congested traffic network is usually highly dependent in terms of link travel times, and thus the assumption of independent link travel times is also in question. These considerations lead us to a more realistic variant, the perfect online information (POI) variant. As stated in the taxonomy, a traveler with perfect online information has knowledge about realizations of all links up to current time. To put it another way, the current-information I is a set $\{C_{jk,t}|(j,k) \in A, t \leq t_0\}$, where t_0 is current time. We will not make specific assumptions about the network statistical dependency. Instead, we will adopt the most general probabilistic description of a network, i.e. the joint realization description, to accommodate all kinds of assumptions on statistical dependency. In particular, a network with strongly dependent link travel times can be handled with this description. It is sometimes a concern that the assumption of perfect online information is not so realistic itself. We acknowledge this, however, as discussed in subsection 2.2.2, variants with perfect online information are easier to study than those with partial online information. Furthermore, in a highly centralized architecture for traffic management, perfect online information is a rather valid assumption. Finally, the algorithm to POI variants also provides building blocks that can be used in developing algorithms for other variants with online information.

In this subsection, we present an operational algorithm DOT-SPI for the perfect on-line information variants. We introduce an important concept of event set, which is a counterpart of current-information in the more general framework and describe the properties of event sets in a POI variant. The general optimality conditions are adopted to the specific case and the algorithm comes out from that naturally. We then proceed with the complexity analysis and point out the importance of finding good approximations for the BRP problems.

2.4.1 Algorithm DOT-SPI

We have a network as described in subsection 2.2.1 with a minor change that the link travel times are *positive*. We seek to find the least expected travel times from all nodes at all departure times with all possible current-information to a certain destination node d. We assume that travelers have perfect on-line information about the link travel times. Mathematically speaking, at any time t, any traveler has knowledge of the realizations of $C_{jk,t'}, \forall (j,k) \in A, \forall t' < t$.

We use a different way to represent the concept of current-information. The current-information defined in the framework of the BRP problem is composed of link travel times. This definition is not convenient for the implementation of the algorithm. At each current time t, each possible joint realization of $C_{jk,t'}, \forall (j,k) \in A, \forall t' < t$, corresponds to a unique set of v_r, therefore we define a new term as the counterpart of current-information in algorithm design. Let $\pi_{jk,t}$ be the realization of $C_{jk,t}$ we have already learned until the current time. Define the event collection $EV := \{v_r | C^r_{jk,t'} =$

$\pi_{jk,t'}, \forall(j,k) \in A, \forall t' < t$, for a certain $t\}$. This is the set of realization candidates after we collect information at time t. As we collect more information (i.e. t increases), the size of EV remains the same or decreases. When EV becomes a singleton, we obtain a deterministic network and can apply any deterministic dynamic shortest path algorithm. Let $\mathbf{EV}(t)$ be the set of all possible event collections at time t and the element of $\mathbf{EV}(t)$ is an event collection $EV = v_r | C^r_{jk,t'} = \pi_{jk,t'}, \forall(j,k) \in A, \forall t' < t$. Specifically, $EV(K-1) = \{\{v_1\}, \{v_2\}, ..., \{v_R\}\}$. All the possible event collections can be generated in preprocessing. Here are some important facts about the event collection:

- There is no overlapping among elements of $EV(t)$ for a given t, so there are at most R event collections at any certain time t ($|EV(t)| \leq R$). Thus there are at most RK event collections in total.
- Any element of $EV(t)$ is a subset of an element of $EV(t-1)$.
- $|EV(t)| \geq |EV(t-1)|$.

A possible scheme of event collections is in Figure 2-15. The rows represents time points in increasing order, i.e. the first row represents the first time point. Each cell in the last row represents a single joint realization v_r, which means that the network becomes deterministic beyond time period $K-1$. At each time t, cells within the bold boundary form an event collection. For example, at time 0, $\{v_1, ..., v_{10}\}$ is one event collection, and $\{v_{11}, ..., v_{15}\}$ is the other. At time 1, when more link travel time realizations are available, $\{v_1, ..., v_{10}\}$ is split into three event collections $\{v_1\}, \{v_2, ..., v_5\}$, and $\{v_6, ..., v_{10}\}$. Other event collections are obtained similarly.

Let $e_{\mu^*}(j, t, EV)$ be the least expected travel time to the destination node d if the departure from node j happens at time t with the event collection EV. Let $\mu^*(j, t, EV)$ be the next arc to take out of node j to realize $e_{\mu^*}(j, t, EV)$. Assume we select arc (j,k) out of node j. At the end of the journey along arc (j,k), we have a new event collection EV' which is one of the possible event collections at time $t + \pi_{jk,t}$. EV' is a random variable and the probability of a certain EV' can be evaluated as following:

$t = 0$	$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}$										$v_{11}, v_{12}, v_{13}, v_{14}, v_{15}$				
$t = 1$	v_1	v_2, v_3, v_4, v_5				$v_6, v_7, v_8, v_9, v_{10}$					$v_{11}, v_{12}, v_{13}, v_{14}, v_{15}$				
$t = 2$	v_1	v_2, v_3, v_4, v_5				$v_6, v_7, v_8, v_9, v_{10}$					v_{11}, v_{12}, v_{13}			v_{14}, v_{15}	
$t = 3$	v_1	v_2, v_3		v_4, v_5		v_6	v_7	v_8, v_9		v_{10}	v_{11}	v_{12}, v_{13}		v_{14}, v_{15}	
$t = 4$	v_1	v_2	v_3	v_4, v_5		v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}, v_{13}		v_{14}, v_{15}	
$t = 5$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}

Figure 2-15: A Possible Scheme of Event Collections

$$Pr(EV'|EV) = \frac{\sum_{r|r\in EV'\cap EV} p_r}{\sum_{r|r\in EV} p_r}, \forall EV' \in \mathbf{EV}(t+\pi_{jk,t}), \forall EV \in \mathbf{EV}(t).$$

Note that $EV' \cap EV = \emptyset$ or EV'.

The optimality conditions for the problem are:

$$\begin{aligned}
& e_{\mu^*}(j,t,EV) \\
= \; & \min_{k\in A(j)}\{\pi_{jk,t} + E_{EV'}[e_{\mu^*}(k,t+\pi_{jk,t},EV')]\} \\
= \; & \min_{k\in A(j)}\{\pi_{jk,t} + \textstyle\sum_{EV\in\mathbf{EV}(t+\pi_{jk,t})} e_{\mu^*}(k,t+\pi_{jk,t},EV') \times Pr(EV'|EV)\}, \forall j \neq d, \\
& e_{\mu^*}(d,t,EV) = 0, e_{\mu^*}(j,t \geq K-1,EV) = e_{\mu^*}(j,K-1,EV) \\
& \forall t \in T, \forall EV \in \mathbf{EV}(t)
\end{aligned}$$

The solution of these equations can be carried out in a decreasing order of time, since the evaluation of $e_{\mu^*}(j,t,EV)$ only depends on $e_{\mu^*}(j,t',EV')$, where $t' > t$. At time $K-1$ or beyond, the network becomes deterministic and static, and we can use any deterministic static shortest path algorithm to compute $e_{\mu^*}(j,t,V), \forall j \in N, \forall t \in K-1, \forall EV \in \mathbf{EV}(K-1)$. Denote the algorithm as DOT-SPI (a counterpart of Algorithm DOT [9] to solve a stochastic problem with perfect information). The statement is as follows.

Algorithm DOT-SPI

Step 0: (Construct $EV(t), t = 0, ..., K-1$)

Call Generate_Event_Collection

```
Step 1: (Initialization)
        1.1 Compute $e_{\mu^*}(j, K-1, EV), \forall j \in N, \forall EV \in \mathbf{EV}(K-1)$
        1.2 $e_{\mu^*}(j, t, EV) \leftarrow +\infty, \forall j \in N\backslash\{d\}$,
            $e_{\mu^*}(d, t, EV) \leftarrow 0$,
            $\forall t < K-1, \forall EV \in \mathbf{EV}(t)$
Step 2: (Main Loop)
        For $t = K-1$ down to 0
            For each $EV \in \mathbf{EV}(t)$
                For each arc $(j,k) \in A$
                    $temp = \pi_{jk,t} +$
                            $\sum_{EV' \in \mathbf{EV}(t+\pi_{jk,t})} e_{\mu^*}(k, t+\pi_{jk,t}, EV') \times Pr(EV'|EV)\}$;
                    If $temp < e_{\mu^*}(j, t, EV)$
                            $e_{\mu^*}(j, t, EV) = temp$
                            $\mu^*(j, t, EV) = k$
```

Generate_Event_Collection

```
$D = \{\{v_1, ..., v_R\}\}$
For $t = 0$ to $K-1$
    For each arc $(j,k) \in A$
        For each disjoint set $S \in D$
            $w$ = number of distinct values among $c^r_{jk,t}, \forall r \in S$;
            Divide $S$ into disjoint sets $S'_1, S'_2, ..., S'_w$,
              such that $c^r_{jk,t}$ is constant over all $r \in S'_i, i = 1, ..., w$ and $\bigcup_i S'_i = S$;
            $D' \leftarrow D'\backslash\{S\} \cup \{S'_1, S'_2, ..., S'_w\}$;
        Next $S$
        $D \leftarrow D'$
    Next $(j,k)$
    $EV(t) \leftarrow D$;
Next $t$
```

2.4.2 Complexity Analysis

The basic step in Generate_Event_Collection is the division of S into disjoint sets. This can be done by sorting the elements in S in time $\theta(slns)$, where s is the cardinality of S. For a given time and a given link, all S are mutually exclusive and collectively exhaustive over all realizations. Assume there are u such disjoint sets for a given time and a given link, $S_1, S_2, ..., S_u$, and $1 \leq u \leq R$. Therefore the sorting of all the u sets takes time $\theta(\sum_{i=1}^{u} s_i lns_i) = \theta(ln \prod_{i=1}^{u} s_i^{s_i}) = O(ln(s_1 + s_2 + ... + s_u)^{s_1+s_2+...+s_u}) = O(RlnR)$. On the other hand, the sorting has to retrieve all the R realizations at least once, so the running time is also $\Omega(R)$. Altogether constructing event collections takes time $O(mKRlnR)$ and $\Omega(mKR)$. Step 1.1 is solving R static shortest path problems, so the running time is $\theta(R \times SSP)$. Step 1.2 takes time $\theta(KRn)$. At a given time t and for a given link (j,k), the evaluation of all $Pr(EV'|EV)$ takes time $\theta(R)$. There are altogether K time periods and m links, so the main loop has a running time of$\theta(mKR)$.

To sum up, Algorithm DOT-SPI has a complexity of $O(mKRlnR+R\times SSP)$ and $\Omega(mKR + R \times SSP)$. This algorithm is strongly polynomial in R, however R could be an exponential function of m. If the link travel times are highly dependent, we expect that R is much less than Qm, where Q is the maximum number of realizations for a single link travel time, but it is still very likely that R is exponential in m. Other variants with less online information could also have running time exponential in number of link travel time random variables involved in current-information.

In fact, this is a well-known drawback of dynamic programming, the so-called Bellman's "curse of dimensionality". Approximations and heuristics of dynamic programming have been a very active research topic in the research community of dynamic programming and stochastic control for a long time and many encouraging results exists [4] [5]. In the transportation community, however, it is believed that no research has been done on systematically designing heuristics and approximations in light of stochastic optimization. The present research serves as a first step in this direction. Future work of this research will largely focus on finding efficient heuristics that perform well in transportation applications.

Chapter 3

Approximations for the BRP Problem in STD Networks

In this chapter, we study approximations to the POI variant studied in Chapter 2. Four approximations are presented with analysis on their efficiency and effectiveness. This analysis is done both theoretically and computationally. The computational tests are not comprehensive, but they provide insights into the performance of approximations. Other approximations are suggested, however without computational tests.

3.1 Four Approximations

3.1.1 The Certainty Equivalent (CE) Approximation

The certainty equivalent approximation is most commonly used in traffic applications. The CE approximation replaces every link travel time random variable by its expected value. Thus it transforms the stochastic network into a deterministic network. It then applies any dynamic shortest path problem algorithm (e.g. Algorithm DOT) to obtain an "optimal" path $p^*(j,t)$. Define $CE(j,t)$ as the expected travel time from node j and departure time t when the path $p^*(j,t)$ is taken. The running time of CE is the same as that of a deterministic dynamic shortest path algorithm, but its solution could be arbitrarily worse than the optimal, as shown by the example in Figure 1-2. The "optimal" path output from CE will be path $a - b$. The expected travel time of path $a - b$ is $6 + M/2$, which could be arbitrarily worse than the expected travel time of the optimal routing policy, which is 10.

3.1.2 The No-Information (NI) Approximation

NI variant can be solved in polynomial time as stated in the complexity analysis of Section 2.3. NI formulation of the BRP problem is valid when the network is time-wise and link-wise independent and the link travel time realizations at the current time are not available. Therefore NI could serve as a good approximation to POI when the statistical dependency of link travel times is weak. Note that NI works with the marginal distributions of link travel times instead of joint distributions. Define $NI(j,t)$ as the expected travel time from node j at departure time t when the routing policy output from the NI approximation is applied. However, the performance of NI as an approximation can also be arbitrarily worse than the optimal. We will not prove this directly. Rather it can be proved as a byproduct of the following statement.

$NI(j,t)$ can be either greater or less than $CE(j,t)$ for a given network. An intuitive argument is that both NI approximation and CE approximation are working on joint distributions distorted from the original one. Which leads to a travel cost farther from the optimal solution depends on the data, as illustrated in the following example.

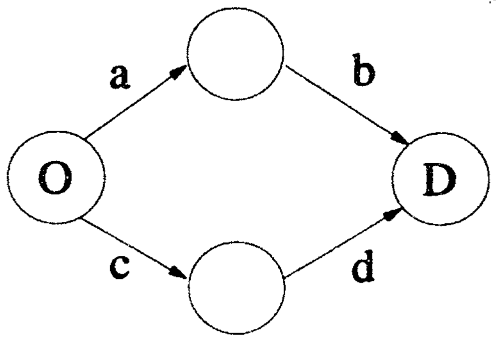


Figure 3-1: CE vs. NI: The Network

Time	Link	v_1	v_2
0	a	1	2
	b	n/a	n/a
	c	1	2
	d	n/a	n/a
1	a	n/a	n/a
	b	1	1
	c	n/a	n/a
	d	1	1
2	a	n/a	n/a
	b	1	3
	c	n/a	n/a
	d	3	1

Joint Realizations
($p_1 = x, p_2 = y = 1 - x$)

Time	Link	Travel Times
0	a	$1(w.p.\ x), 2(w.p.\ y)$
	b	n/a
	c	$1(w.p.\ x), 2(w.p.\ y)$
	d	n/a
1	a	n/a
	b	$1(w.p.\ 1)$
	c	n/a
	d	$1(w.p.\ 1)$
2	a	n/a
	b	$1(w.p.\ x), 3(w.p.\ y)$
	c	n/a
	d	$3(w.p.\ x), 1(w.p.\ y)$

Marginal Distributions
($x + y = 1$)

Table 3.1: CE vs. NI: Travel Times

The network in Figure 3-1 has two possible joint realizations of all link travel times. The corresponding marginal PMF is also provided. The expected link travel times are not listed, as they can be easily computed from the marginal PMF. There is only one O-D pair, and we only study departure time 0. The expected travel time of path $a-b$ is $x(1+1)+y(2+3) = 2x+5y$, and that of path $c-d$ is $x(1+1)+y(2+1) = 2x+3y$. As y is positive, the expected travel time of path $a-b$ is greater than that of path $c-d$.

Now let us see how the NI approximation and CE approximation will make the routing decisions. When NI approximation is applied, we work on the marginal distribution instead of the joint distribution. The "expected travel time" of path $a-b$ computed from NI would be $x(1+1)+y(2+x+3y)$, and that of path $c-d$ would be $x(1+1)+y(2+3x+y)$. The difference between expected travel times of path $a-b$ and path $c-d$ in NI is $2y(y-x)$. Note that in this example, the routing policy from NI reduces to paths, due to the special topology of the network.

Let $x = 3/4, y = 1/4$, then NI will choose path $a-b$. Assume the travel time of link b at time $5/4$ is greater than the travel time of link d at time $5/4$, then CE will choose path $c-d$. In this case, CE is better than NI. Let $x = 1/4, y = 3/4$, then NI will choose path $c - d$. Assume the travel time of link b at time 7/4 is less than the travel time of link d at time 7/4, then CE will choose path $a - b$. In this case, NI is better than CE. Note that we use fractions in departure times only to minimize the efforts in presenting data. Actually one can always multiply the existing data by a large enough number to obtain integral data.

Since NI can have worse solutions than CE, and CE can have solutions that are arbitrarily worse than the optimal, NI also can have solutions that are arbitrarily worse than the optimal.

3.1.3 The Open Loop Feedback with Certainty Equivalent Approximation (OLFCE)

OLFCE is an improved certainty equivalent approximation. At each decision node, travelers employ a CE that replaces every link travel time random variable in later times by its expected value conditional on the network conditions realized so far. Travelers follow the resulted "optimal" path until a new decision node is reached. At that time, a CE is applied again, conditional on the updated network conditions. Define $OLFCE(j, t)$ as the expected travel time from node j and departure time t when the series of "optimal" paths generated by the open loop feedback with CE approximation are followed. It was proved in by Bertsekas [4] that "open loop feedback controls perform at least as well as open loop controls" in dynamic programming. This result can be translated into the terminology of this thesis such that routing policies generated by OLFCE performs at least as well as a minimum expected travel time path. Since output from CE are paths and cannot perform better than the minimum expected travel time paths, we have $OLFCE(j, t) \leq CE(j, t)$. The running time of the OLFCE is $min(K, R)$ times the time to solve one CE, and still the performance of OLFCE can be arbitrarily worse than the optimal. To obtain an example to show this, we can set the conditional link travel times as those used to prove the performance of CE could be arbitrarily worse than the optimal.

3.1.4 The Open Loop Feedback with No-Information Approximation (OLFNI)

Both CE and OLFCE have appeared in the literature for quite a long time. The transition from CE to OLFCE suggests a new approximation OLFNI developed from the NI approximation. Similar to OLFCE, at each decision node, travelers employ an NI approximation that works on the marginal distributions of link travel times conditional on the network conditions realized so far. Travelers follow the resulted "optimal" routing policy until a new decision node is reached. At that time, an NI approximation is applied again, conditional on the updated network conditions. It is conjectured that OLFNI will perform at least as well as NI. However, its performance can still be arbitrarily worse than the optimal.

Similar to the relationship between CE and NI, results from OLFCE could be either greater or less than results from OLFNI. To obtain an example to show this, we can set the conditional link travel times as those used to prove $CE(j,t)$ can be either greater or less than $NI(j,t)$.

3.1.5 Theoretical Study of DOT-SPI vs. Approximations

We define $POI(j,t)$ as the expected travel time from node j at departure time t when the optimal routing policy obtained from Algorithm DOT-SPI is applied. Any routing policy generated by CE, NI, OLFCE, or OLFNI is a feasible routing policy for the perfect online information variants. For example, the routing policy generated by CE can be viewed as a routing policy in the POI variant, such that $\mu^*(j,t,EV)$ is constant over all $EV \in \mathbf{EV}(t)$, and all $t \in T$. The routing policy generated by NI can also be viewed as a routing policy in the POI variant, such that $\mu^*(j,t,EV)$ is constant over all $EV \in \mathbf{EV}(t)$. As Algorithm DOT-SPI solves the POI variant, $POI(j,t)$ is no greater than any one of $CE(j,t), NI(j,t), OLFCE(j,i), OLFNI(j,t), \forall j \in N, \forall t \in T$.

3.2 Computational Tests

There is a trade-off between effectiveness and efficiency for all approximations, i.e. they could have satisfactory running times, but their results could be arbitrarily worse in absolute value than those obtained by running the exact algorithm. The effectiveness of approximations largely depends on specific applications.

In this section, computational tests are designed to study the effectiveness of the four approximations presented in the previous subsection. Algorithms and approximations are run on randomly generated networks. The optimal results from Algorithm DOT-SPI are used as a benchmark. The percent relative difference between approximation results and Algorithm DOT-SPI results is used as the measure of effectiveness. Various parameters that may affect the relative difference are checked. Due to the tremendous computational burden, the results presented in this section are only preliminary. Further tests would need to be done to test the approximations in larger varieties of networks and with border ranges of parameters.

3.2.1 The Random Network Generator

The computational tests are conducted on randomly generated networks. A multivariate normal distribution is assumed for the joint distribution of all link travel time random variable. The random network generator takes as input: 1) the number of nodes, 2) the number of links, 3) the number of time periods, 4) the number of realizations, 5) the homogeneous link travel time mean, 6) the homogeneous standard deviation of link travel times, 7) the homogeneous correlation coefficient of link travel times, 8) the maximum in-degree, and 9) the maximum out-degree.

The topology of the network is randomly generated. The last node is the default destination node. An in-tree rooted at the destination node is generated to ensure the connectivity to the destination node. The remaining links are generated randomly, respecting the maximum in-degree and out-degree.

The joint realizations of all link travel times are generated by a routine that can generate samples from a multivariate normal distribution. The number of random variables is the number of links times the number of time periods. A homogeneous mean travel time and a homogeneous standard deviation are used for every link travel time random variable. A homogeneous correlation coefficient is used for every pair of random variables. The standard deviation should be carefully chosen so that most of the sample values are positive. In the case that a negative value is generated, the absolute value is taken. When the link travel times are read by algorithms or approximations, they are rounded to the nearest integers. The probability of each joint realization is obtained by first generating R numbers between 0 and 1, and then normalizing the R numbers by their sum.

Sampling from multivariate normal distribution is very time-consuming, so our network sizes are restricted. Despite the limited sizes, these tests can provide insights into the performance of approximations.

3.2.2 The Measure of Effectiveness

We study the percent relative difference between results from Algorithm DOT-SPI and results from the four approximations. The definition of the percent relative difference is as follows:

$$\Delta_{approximation} = \frac{\sqrt{\sum_{i=0}^{K-1}\sum_{j=1}^{N}(POI(j,t) - approximation(j,t))^2}}{\sqrt{\sum_{i=0}^{K-1}\sum_{j=1}^{N}(POI(j,t))^2}}$$

where *approximation* can take the value CE, NI, OLFCE, or OLFNI. In the computation of percent relative differences, the weight of each (j,t) pair is the same. This implies that we assume the demand to the destination node is distributed evenly across both space and time.

We study the magnitude of the percent relative differences as a function of four different parameters: the homogeneous standard deviation of link travel times, the homogeneous correlation coefficient of link travel times, the number of realizations, and the average in- and out-degrees.

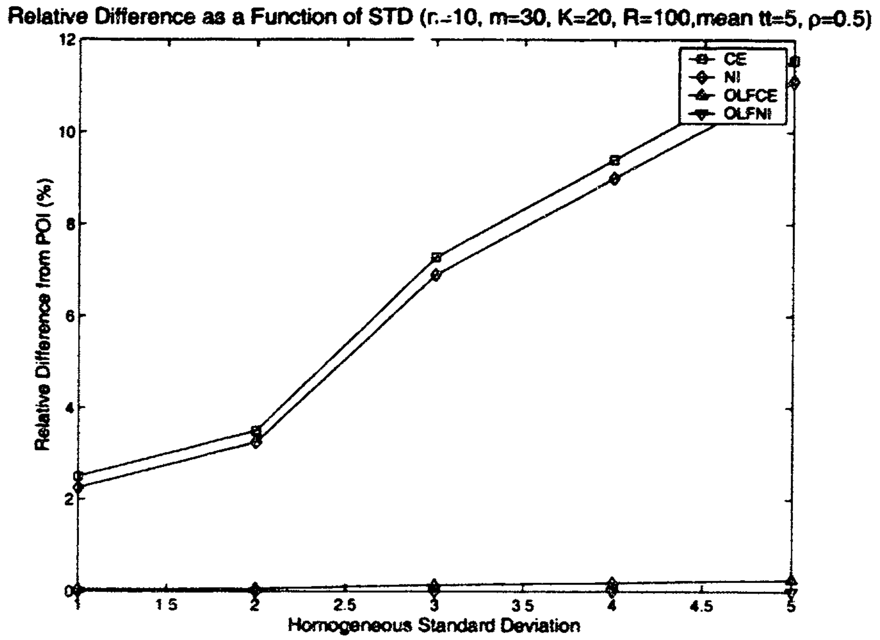
The implemented approximations are just proxy of their real-life counterparts. For example, the expected link travel times for CE or OLFCE should come directly from observation or other estimation methods, not from taking expectation over the joint realizations. Similarly, the marginal distributions of link travel times for NI or OLFNI should also come directly from observation or other estimation methods, rather than from aggregating joint realizations. The possible bias between the observed expected link travel times (marginal distributions) and the computed ones from the joint realizations may further complicate the assessment of performance of approximations.

3.2.3 Tests Design and Results

Algorithm DOT-SPI and approximations CE, NI, OLFCE, OLFNI are implemented in C++ and run on a Dell OptiPlex GX110 workstation with 933 MHz CPU speed and 256 megabytes RAM and under Red Hat Linux 7.0. A test is defined as obtaining results from the five implemented algorithm/approximations for a given combination of the input data to the random network generator. Each test is composed of ten identical runs. The average over the ten runs are taken as the result of this particular test.

The results of the tests are shown in Figure 3-2 through Figure 3-5. The upper graph in each figure shows the results for all the four approximations. The lower graph in each figure shows the results for only the two open loop feedback approximations, as they are different in scale from the other two. There are some general observations for all the tests. The magnitude of the percent relative difference for CE and NI is around 10, and that for OLFCE and OLFNI is very close to zero. This supports the arguments that OLFCE always performs better than CE and OLFNI always performs better than NI. The performance of the two open loop feedback approximations are very close to that of Algorithm DOT-SPI, partly due to the small number of joint realizations. When the number of joint realizations is small, the value of the online information is large and the network becomes deterministic very soon after the starting time point. Since in a deterministic network, CE, NI and Algorithm DOT-SPI give the same expected travel times, it is not surprising that OLFCE and OLFNI have very close results as Algorithm DOT-SPI in this situation. Another interesting observation is that generally NI performs better than CE and OLFNI better than OLFCE, although the theoretical study shows that CE could perform better than NI and OLFCE could perform better than OLFNI. This is not surprising, as NI outputs routing policies that make use of the information on arrival times, while CE outputs paths that totally ignore any information one may obtain online. It is conjectured that when the average number of possible next nodes is small, the performance of CE and NI is close, as shown in the example of Figure 3-1, since routing policies would nearly reduce to paths in this situation.

Figure 3-2 shows the percent relative difference as an increasing function of the homogeneous standard deviation of link travel times. Note the mean link travel time is also homogeneous across time and space. When the standard deviation is large, the link travel times are more dispersed, and thus the expected travel times of different paths (routing policies) are more likely to be apart from each other. This magnifies the difference between optimal and sub-optimal solutions. Figure 3-3 shows the percent relative difference as a decreasing function of the homogeneous correlation coefficient of link travel times. This phenomenon can be explained by the same logic used in Figure 3-2. A positive correlation coefficient of random variables X and Y provides a measure of extent to which the signs of $x - E[X]$ and $y - E[Y]$ "tend" to be positive. As we have a homogeneous mean for all link travel times, the positive correlation coefficient actually indicates how close x and y are. When link travel times are close, the difference between optimal and sub-optimal solutions is reduced. Figure 3-4 shows the percent relative difference as a function of the number of realizations. The number of realizations represents, among others, the extent of discretization. There is no definite relationship shown in the figure. Further computational tests are needed to study the effect of discretization in a larger range. Figure 3-5 shows the percent relative difference as an increasing function of average in-degree and out-degree. The two degrees are set to be equal in the tests. As the average degree increases, the travelers have more choices of the next node. Therefore more paths are involved in an optimal routing policy, and the optimal routing solutions have more chance to achieve lower travel times than the sub-optimal solutions.



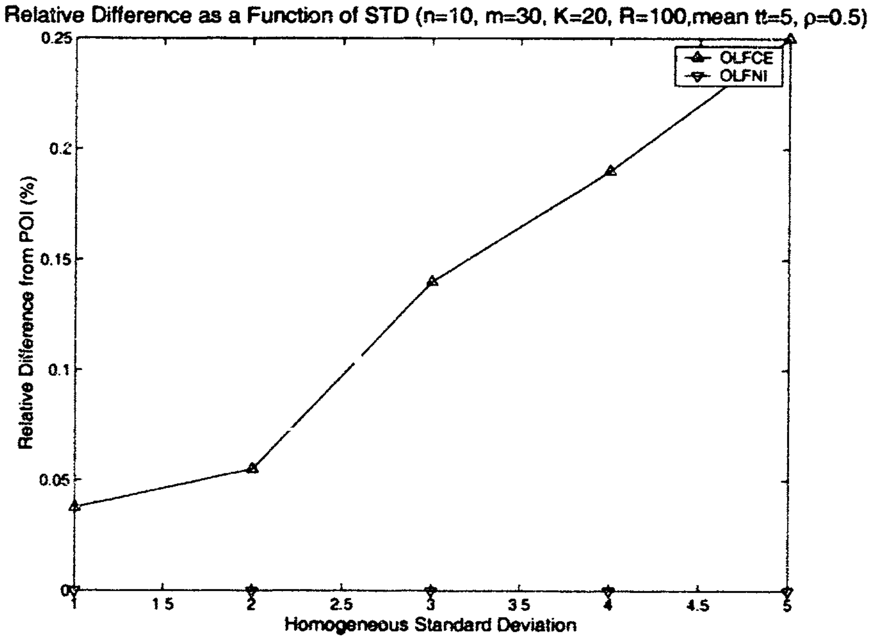


Figure 3-2: Percent Relative Difference as a Function of the Homogeneous Standard Deviation of Link Travel Times (with 10 nodes, 30 links, 20 time periods, 100 joint realizations, 10 as the homogeneous mean link travel time, and 0.5 as the homogeneous correlation coefficient of link travel times

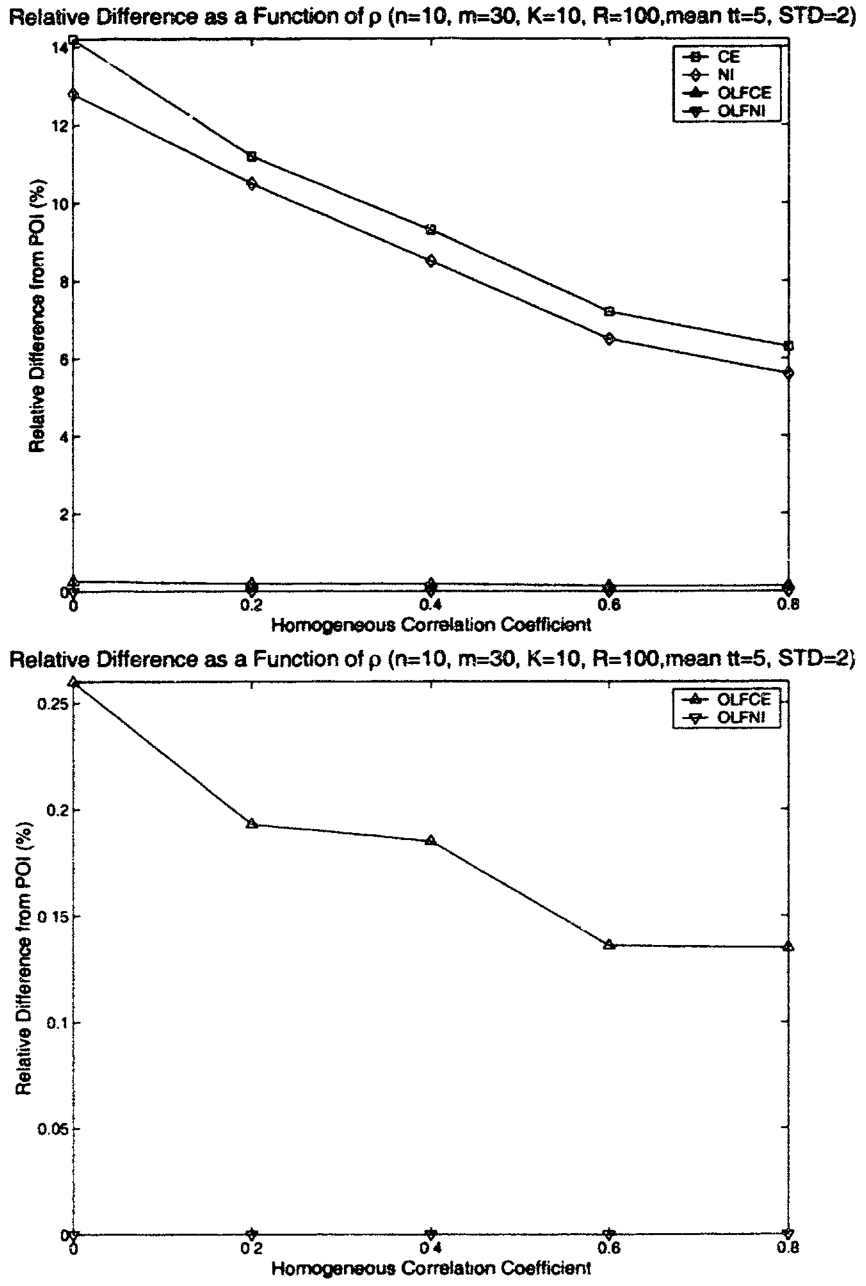


Figure 3-3: Percent Relative Difference as a Function of the Homogeneous Correlation Coefficient of Link Travel Times (with 10 nodes, 30 links, 10 time periods, 100 joint realizations, 5 as the homogeneous mean link travel time, and 1 as the homogeneous standard deviation of link travel times

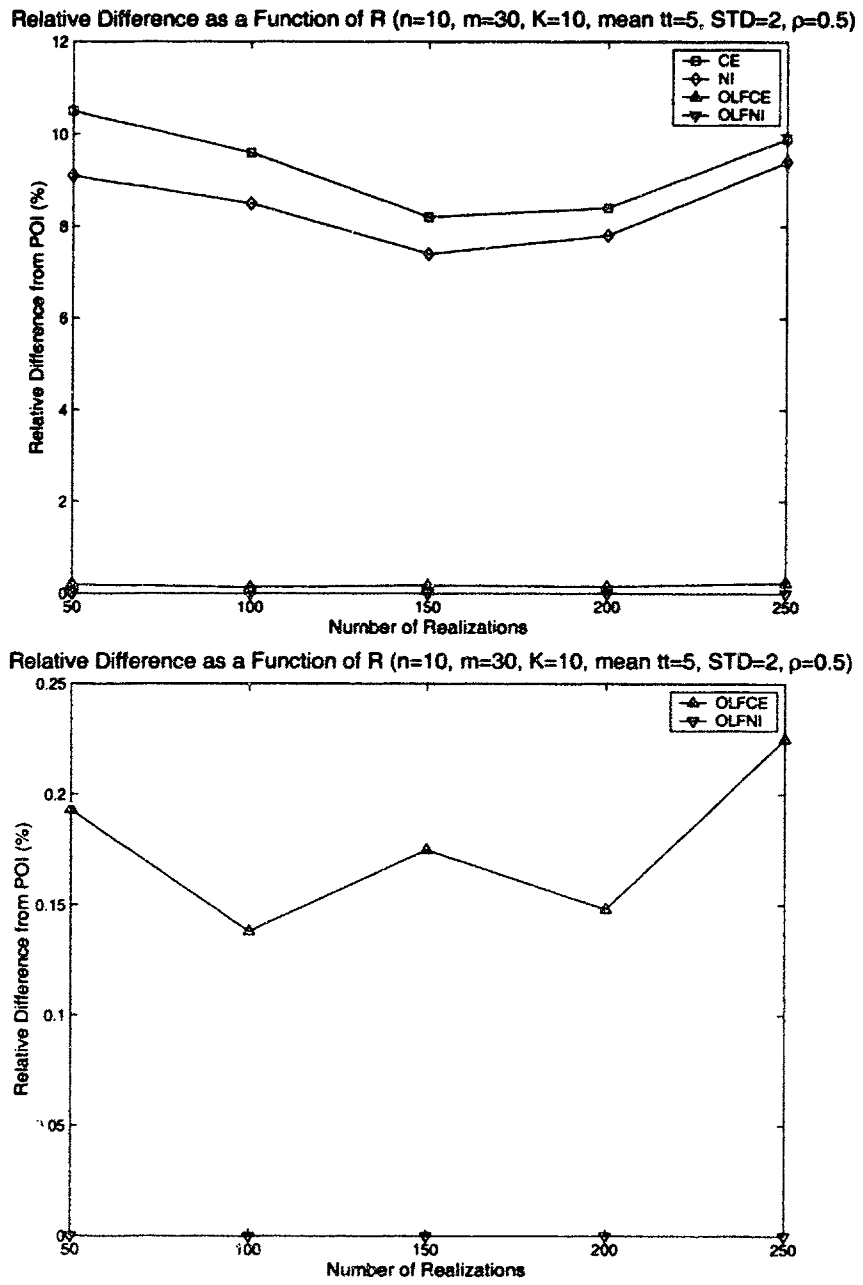
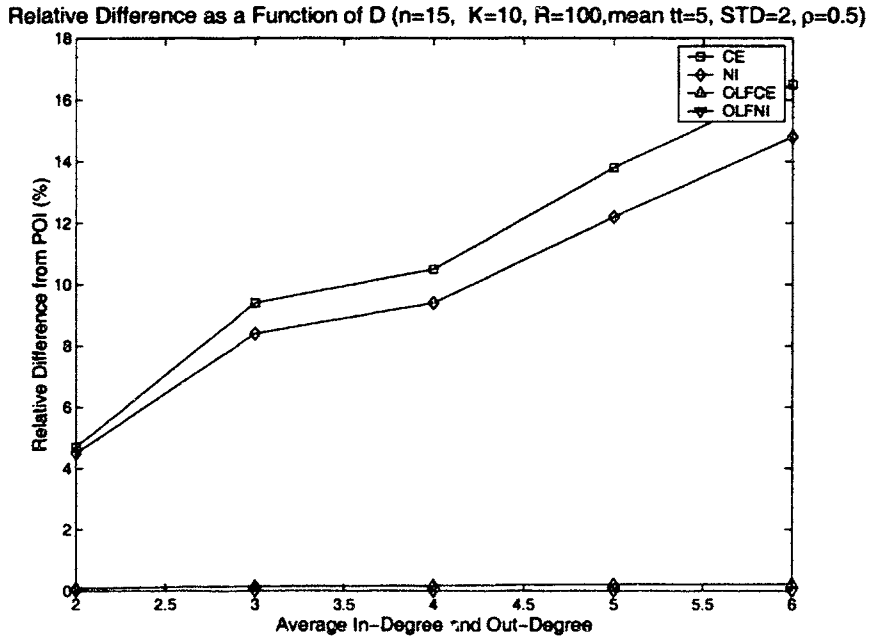


Figure 3-4: Percent Relative Difference as a Function of the Number of Joint Realizations (with 10 nodes, 30 links, 10 time periods, 5 as the homogeneous mean link travel time, 2 as the homogeneous standard deviation of link travel times, and 0.5 as the homogeneous correlation coefficient of link travel times



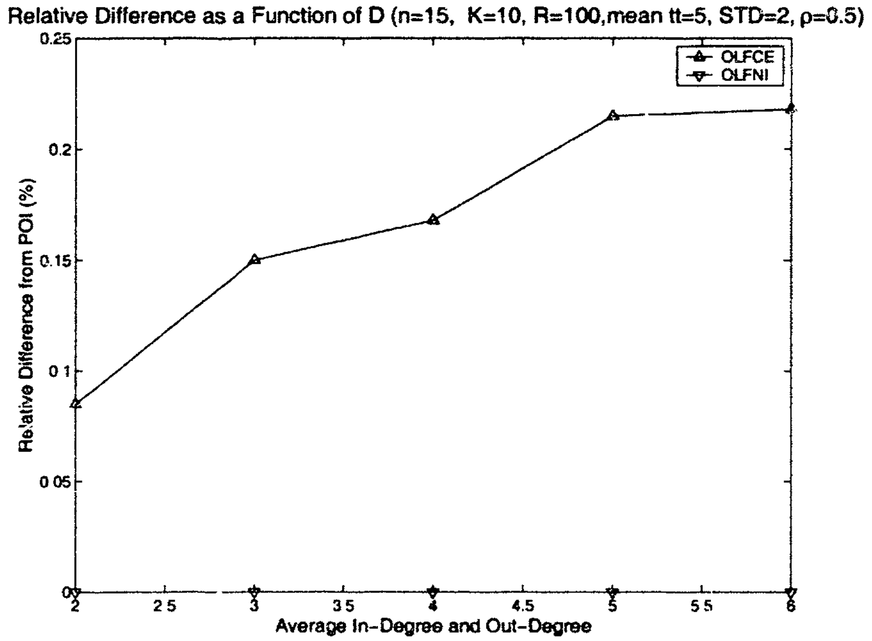


Figure 3-5: Percent Relative Difference as a Function of the Average In-Degree and Out-Degree (with 15 nodes, 10 time pericds, 100 joint realizations, 5 as the homogeneous mean link travel time, 2 as the homogeneous standard deviation of link travel times, 0.5 as the homogeneous correlation coefficient of link travel times, and 2 times the average in- and out-degree as the maximum in- and out-degree

Chapter 4

A Policy-Based Stochastic Dynamic Traffic Assignment Model

4.1 Introduction

Dynamic Traffic Assignment (DTA) methods constitute parts in the intelligent core of Intelligent Transportation Systems (ITS). They provide support to the design, evaluation, operation of Advanced Traffic Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS). A DTA model captures the interaction between traffic demand and network supply in a time-dependent context and aims to estimate and/or predict network conditions, such as link travel times, O-D travel times, and link volumes, to support traffic management decision making and travelers' information provision. It is natural that one of the critical requirements of a DTA model is its accuracy in estimating/predicting traffic conditions.

Stochasticity in transportation systems is both intuitively prevalent and experimentally proven, as discussed in 1.1. Therefore there is a need to capture stochasticity in DTA models and to study its implications and significance of stochasticity in DTA methods.

Over the years, there have been various approaches to introduce stochasticity in traffic assignment models. Early developments addressed by stochasticity in **static** traffic assignment methods. Daganzo and Sheffi [15] established the Stochastic User

Equilibrium (SUE), where users have random perception errors of the true travel costs. The resulting path choices are therefore naturally random, in the form of path choice probability. Equilibrium path flows, however, are not presented as distributions. Instead, a "large sample" approximation is used, such that the proportion of travelers that take a given path "equals" its probability to be chosen by an individual traveler. Consequently, an "average" deterministic flow pattern is obtained.

Two later papers extended Daganzo and Sheffi's work in two different directions of considering stochasticity. Mirchandani and Soroush [22] considered the case where stochasticity comes from both traveler perception errors and link travel costs themselves. A model is developed to consider the travelers' risk taking behavior, which is an important factor to take into account in the presence of stochasticity. The "large sample" approximation is also used in their work. The other extension is described in Hazelton [18], which addressed stochasticity emanating from the same source as in [15]. The key new idea developed in [18] is the representation of SUE route choice conditions as a joint probability distribution, defined as the conditional route choice of each individual given the choices of other travelers. The equilibrium conditions are presented using what Hazelton [18] terms as the Conditional Stochastic User (CSU) behavior conditions. A numerical example is used to show that the equilibrium expressed by CSU gives different results from the equilibrium as a "large sample" approximation developed in [15]. The CSU and SUE tend towards each other as teh demand increases, as is expected.

Cantarella and Cascetta[7] [8] studied traffic equilibrium from the point view of fixed point attractors. The stochasticity considered comes from traveler perception errors. The process of achieving equilibrium is viewed as a continuous interplay between demand and supply, which can be modeled as a deterministic or as a stochastic process. Both a day-to-day and a with-in day contexts are considered. In particular, the conditions for the existence and uniqueness of a stationary stochastic process are provided and their relationship to an equilibrium state is studied.

The differences between Hazelton[18] model and Cantarella and Cascetta [7] [8] model are mainly the following: 1) Hazelton's model is expressed in an atemporal framework, thus the Conditional Stochastic User Behavior was devised to solve the paradox in defining the conditional distribution of route choices. Canterella and Casetta's model actually has a temporal dimension and thus is not subject to the paradox. 2) Hazelton's CSUE model is obtained by solving a system of equations that represent the equilibrium conditions, while Canterella and Casetta's model suggests an iterative algorithm that simulate the convergence to a fixed-point in the form of a stationary distribution.

All the above papers dealt with static traffic assignment. In the research body on dynamic traffic assignment, stochasticity is usually incorporated through a direct extension of SUE of Daganzo and Sheffi [15]. That is, the proportion of travelers that choose a certain path between a given O-D pair at each time interval is deemed as the probability that the path is chosen by an individual traveler for the corresponding time interval. Peeta and Zhou [23] proposed a hybrid framework for on-line dynamic traffic assignment in consideration of demand/supply stochasticity. The framework is composed of DTA solutions generated off-line and then adjustmented on-line. The demand stochasticity is taken account of at the off-line stage. A set of realizations of time-dependent O-D trips is generated from the historical database, and a traditional DTA algorithm is run for each realization to obtain path flow assignments for that realization. The expected path flow assignments are calculated by computing the expectation of results obtained over all the O-D realizations. The online component then uses the expected path flows as an initial solution and make adjustment according to unfolding conditions, e.g. actual O-D trips and/or incidents. Robustness of the off-line solutions are then studied.

In summary, stochasticity in dynamic traffic assignment models is mostly considered as from traveler perception errors. No papers in the literature have considered DTA models that work with general time-dependent link travel time distributions. On the other hand, outputs from DTA models in the literature are "average" values of network variables rather than their distributions. Furthermore, all DTA models are based on paths. However, as we have studied in the previous two chapters, routing policies can lead to less expected travel times in STD networks.

In this chapter, we aim to build a dynamic traffic assignment model that works with general link travel time distributions. This approach allows for a better representation of stochasticity in traffic modeling. Our model is policy-based, with the expectation that it can give different results from those of path-based models. The outputs are link travel time distributions, which allows for a richer richer representation of traffic. Better traffic managment decisions are then possible based on this richer representation.

This chapter is organized as follows. In Section 4.2, a conceptual framework for the policy-based stochastic DTA model is introduced. In Section 4.3, the conditions for a policy-based stochastic DTA model are presented. In Section 4.4, we give an illustrative example to show how the policy-based DTA model can work and to explore some unique properties of the model. This is to provide the reader with intuitive understanding of a policy-based model. More rigorous developement are given in Section 4.6 through Section 4.8. In Section 4.5, we provide all the notation needed for the development of the model. In Section 4.6, the users' routing choice model is established. The dynamic network loading model is developed with the consideration of queues in Section 4.7. Finally we propose in Section 4.8 the solution algorithms for the users' policy choice model, the dynamic network loading model and a heuristic DTA algorithm is proposed.

4.2 A Conceptual Framework for the Policy-Based Stochastic DTA Model

A conceptual framework for this model is shown in Figure 4-1. The input is the time-dependent O-D trips and the stochastic dynamic supply, such as the probability of incidents, the random length of the duration of an incident, or the probability of bad weather. The output is sample distributions from link travel times and other measures of effectiveness of interest (such as the link volume and O-D travel times) and the corresponding routing policy flows. There are three major components of the stochastic DTA model:

- the users' policy choice model,
- the dynamic network loading model,
- and the routing policy generation model.

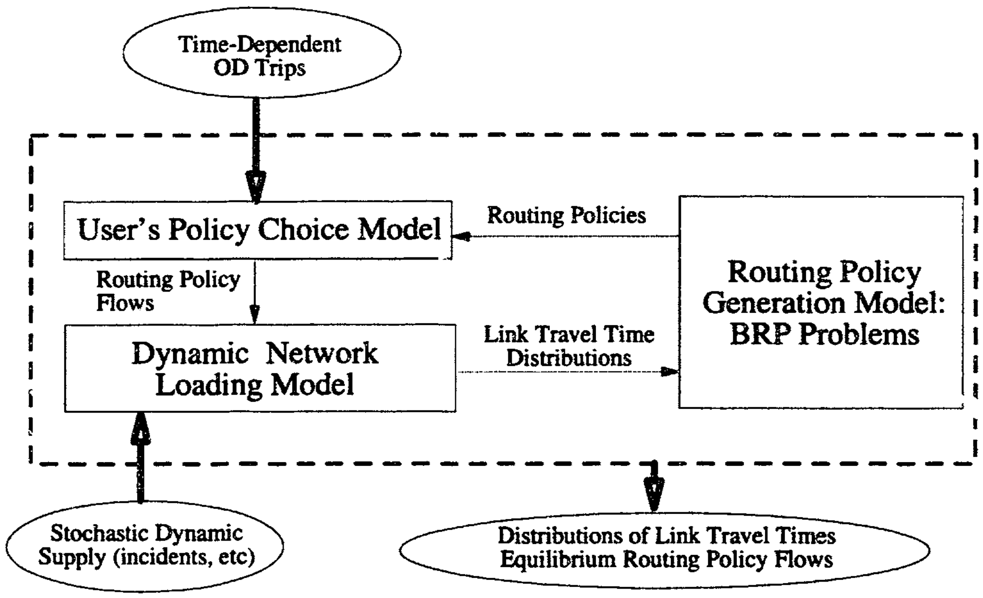

Time-Dependent OD Trips
User's Policy Choice Model
Routing Policies
Routing Policy Flows
Routing Policy Generation Model: BRP Problems
Link Travel Time Distributions
Dynamic Network Loading Model
Stochastic Dynamic Supply (incidents, etc)
Distributions of Link Travel Times Equilibrium Routing Policy Flows

Figure 4-1: A Conceptual Framework of Stochastic Dynamic Traffic Assignment Model

The users' policy choice model takes routing policies and the time-dependent demand as input. In discrete time representation, the demand is given as a matrix of time-dependent number of O-D trips during all time intervals. Note the OD matrix is deterministic as in traditional DTA models. For each output from the policy choice model are flows assigned to all routing policies. The policy flows are then loaded to the network by the dynamic network loading model, with stochastic dynamic supplies. The stochastic supplies can be in many forms depending on the application. For example, if we are to do rerouting after an incident occurs, the joint distribution of the duration and the severity of the incident is the stochastic part of the supply. The network loading model is deterministic in itself, as it works with a single realization of network supplies. Multiple loadings are performed with samples from the dynamic stochastic supplies to obtain link travel time sample distributions. The routing policy generation model then takes as input the link travel time sample distributions and produces optimal routing policies. We have studied in details the routing policy generation model, i.e. the best routing policy problem in a stochastic time-dependent network, in Chapter 2 and Chapter 3. In the rest of the chapter, we elaborate on the other two components and build a stochastic DTA model based on these development.

4.3 A Policy-Based Stochastic Dynamic Traffic Assignment Model

According to the modeling framework, the policy-based DTA model contains a users' policy choice model, a dynamic network loading model, and a routing policy generation model, along with the interaction between them. The equilibrium condition generalized from the conventional path-based user-optimal condition is as follows:

> *For each O-D pair at each instant of time, the expected travel time of the used policies by the users departing at the same time are equal and minimal.*

4.4 An Illustrative Example of the Policy-Based DTA

In this section, we use an example to show how the policy-based DTA assignment can work and to show some properties of policy-based DTA results. This example is not aimed to be a realistic representation of actual traffic applications. Rather the data is designed to show key properties of policy-based DTA models.

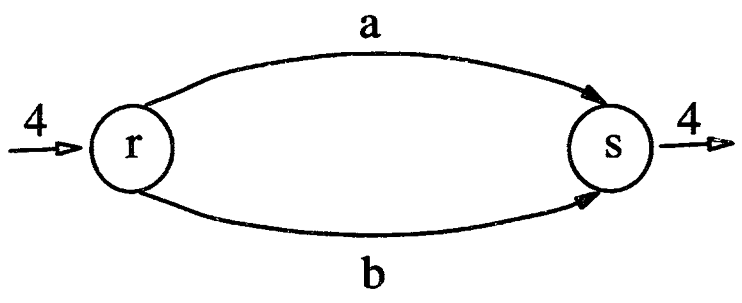


Figure 4-2: An Illustrative Example for DTA

4.4.1 Example Description

The network in Figure 4-2 has a single O-D pair connected by two parallel links. An incident could happen on link a. If no incidents have occurred yet, there is a probability p that the incident will happen in the next time interval. If an incident has happened and the link capacity has been reduced, it will continue with the reduced capacity at the next time interval. In fact, the first time interval k that the incident happens takes a geometric distribution of parameter p. The volume-delay function of link a is accordingly defined as follows.

$$C_a(k)(g_a(k)) = \begin{cases} \begin{cases} g_a(k), & w.p.\ 1-p \\ 5g_a(k), & w.p.\ p \end{cases} , & \text{if no incident happened in the past} \\ 5g_a(k), & \text{if an incident happend at time } k \end{cases}$$

The volume-delay function of link b is defined as follows:

$$C_b(k)(g_b(k)) = 2g_b(k) + 4$$

where $C_a(k)(g_a(k))/C_b(k)(g_b(k))$ denote link travel time of link a/b at time k as a function of link flow rate g_a/g_b at time k.

The time intervals in interest are only time 1 and time 2. The O-D trip rate for both time intervals is 4. All users are risk neutral. Users have information on whether an incident happened in the past, but not at the current time. For example, at time 2, users know whether the incident happened at time 1. Here is a summary of possible policies at time 1 and time 2:

Policy	Time 1	Time 2
μ_1	link a	link a
μ_2	link b	link b
μ_3		$\begin{cases} \text{link } a, & \text{if incident does not happen at time 0;} \\ \text{link } b, & \text{otherwise} \end{cases}$
μ_4		$\begin{cases} \text{link } b, & \text{if incident does not happen at time 0;} \\ \text{link } a, & \text{otherwise} \end{cases}$

Table 4.1: Possible Policies

4.4.2 A Solution to the DTA Problem

The policy-based DTA equilibrium conditions can be expressed by a system of equations and inequalities, as in traditional DTA. In order to obtain a solution to the example problem by solving the system, we need an approach to compute the expected travel time of a routing policy. We obtain the expressions of expected travel times for policies by enumerating all possible situations at a given time. The detailed description is as follows.

The state of link a at time 1 has the following distributions in terms of capacities, along with the relationship between link flows and policy flows at time 1:

Realization	Time 1	Probability	$g_a(1)$	$g_b(1)$
1	reduced capacity	p	$f_1(1)$	$f_2(1)$
2	normal capacity	$1-p$	$f_1(1)$	$f_2(1)$

Table 4.2: States of Link a at Time 1

The decisions at Time 2 requires the knowledge of the state of link a at Time 1. Therefore we have the distribution of states of link a at time 1 AND time 2 as follows, in terms of capacities. The specific forms of policy 3 and policy 4 and the relationship between link flows and policy flows at time 2 are also shown.

Realization	Time 1	Time 2	Probability	μ_3	μ_4
1	reduced	reduced	p	link b	link a
2	normal	reduced	$p(1-p)$	link a	link b
3	normal	normal	$(1-p)^2$	link a	link b

Table 4.3: States of Link a at Time 2 (Part 1)

Realization	Time 1	Time 2	$g_a(2)$	$g_b(2)$
1	reduced	reduced	$f_1(2)+f_4(2)$	$f_2(2)+f_3(2)$
2	normal	reduced	$f_1(2)+f_3(2)$	$f_2(2)+f_4(2)$
3	normal	normal	$f_1(2)+f_3(2)$	$f_2(2)+f_4(2)$

Table 4.4: States of Link a and at Time 2 (Part 2)

There is some additional notation. $f_i(k)$ denote the policy flow of policy i at time k. $g_j^r(k)$ denote link flow of link j at time k in the r^{th} realization, $r = 1, 2$, when $k = 1$, and $r = 1, 2, 3$, when $k = 2$. $C_i(k)$ denotes the expected travel time of policy i at time k.

We solve the system of equations and inequalities by trial-and-error. We first assume all policies are used. If the resulting system has a solution, it is the equilibrium solution. If not, we can assume only some of the policies are used. If a solution is obtained, we can continue to check if the unused policies have expected travel times no less than the used ones. If the answer is yes, we can claim that the solution is the equilibrium solution. The following is the system when all policies are used.

At time 1:

Flow conservation

$$f_1(1) + f_2(1) = 4$$

Equilibrium condition

$$C_1(1) = C_2(1)$$

Expression of expected travel times of μ_1 and μ_2

$$\begin{aligned}
C_1(1) &= p \times 5g_a^1(1) + (1-p) \times g_a^2(1) \\
C_2(1) &= p \times (2g_b^1(1) + 4) + (1-p) \times (2g_b^2(1) + 4)
\end{aligned}$$

Relationship between link flows and policy flows

$$\begin{aligned}
g_a^1(1) &= f_1(1) \\
g_a^2(1) &= f_1(1) \\
g_b^1(1) &= f_2(1) \\
g_b^2(1) &= f_2(1)
\end{aligned}$$

Nonnegativity constraints

$$\begin{aligned}
f_1(1) &\geq 0 \\
f_2(1) &\geq 0
\end{aligned}$$

At time 2:

Flow conservation

$$\textstyle\sum_{i=1}^4 f_i(2) = 4$$

Equilibrium condition

$$\begin{aligned}
C_1(2) &= C_2(2) \\
C_2(2) &= C_3(2) \\
C_3(2) &= C_4(2)
\end{aligned}$$

Expression of expected travel times of $\mu_1, \mu_2, \mu_3, \mu_4$

$$\begin{aligned}
C_1(2) &= p \times 5g_a^1(2) + p(1-p) \times 5g_a^2(2) + (1-p)^2 \times g_a^3(2) \\
C_2(2) &= p \times (2g_b^1(2) + 4) + p(1-p) \times (2g_b^2(2) + 4) + (1-p)^2 \times (2g_b^3(2) + 4) \\
C_3(2) &= p \times (2g_b^1(2) + 4) + p(1-p) \times 5g_a^2(2) + (1-p)^2 \times g_a^3(2) \\
C_4(2) &= p \times 5g_a^1(2) + p(1-p) \times (4g_b^2(2) + 4) + (1-p)^2 \times (4g_b^3(2) + 4)
\end{aligned}$$

Relationship between link flows and policy flows

$$\begin{aligned} g_a^1(2) &= f_1(2) + f_4(2) \\ g_a^2(2) &= f_1(2) + f_3(2) \\ g_a^3(2) &= f_1(2) + f_3(2) \\ g_b^1(2) &= f_2(2) + f_3(2) \\ g_b^2(2) &= f_2(2) + f_4(2) \\ g_b^3(2) &= f_2(2) + f_4(2) \end{aligned}$$

Nonnegativity constraints

$$\begin{aligned} f_1(1) &\geq 0 \\ f_2(1) &\geq 0 \\ f_3(1) &\geq 0 \\ f_4(1) &\geq 0 \end{aligned}$$

The equations corresponding to time 1 and time 2 are actually decoupled, so we can solve them separately. An Excel solver is used to solve the system and a solution is found. The solution includes the policy flows ($f_1(1)$, $f_2(1)$, $f_1(2)$, $f_2(2)$, $f_3(2)$, $f_4(2)$), the link flow distributions ($g_a^1(1)$, $g_a^2(1)$, $g_b^1(1)$, $g_b^2(1)$, $g_a^1(2)$, $g_a^2(2)$, $g_a^3(2)$, $g_b^1(2)$, $g_b^2(2)$, $g_b^3(2)$), and the equilibrium (minimum) expected policy travel times ($C_1(1) = C_2(1)$, $C_1(2) = C_2(2) = C_3(2) = C_4(2)$). We can obtain link travel time distributions from the link flow distributions using the expressions of expected travel times of policies.

We should not conclude that the assignment policy flows are unique. If we set one or more of the policy flows to be zero and eliminate corresponding equations, we obtain another system of equations that could also possibly satisfy the equilibrium condition and that can have solutions. Our computation shows that there are indeed multiple solutions for this example. For example, when $p = 0.1$, the equilibirium is attained both when all policies are used or only μ_1, μ_3, μ_4 are used.

4.4.3 Solve the Problem with MSA

For realistic applications, it might be very time-consuming to solve directly the system of equilibrium conditions. We use the MSA method [27] to solve this example. The convergence resuls are shown below for $p = 0.1$.

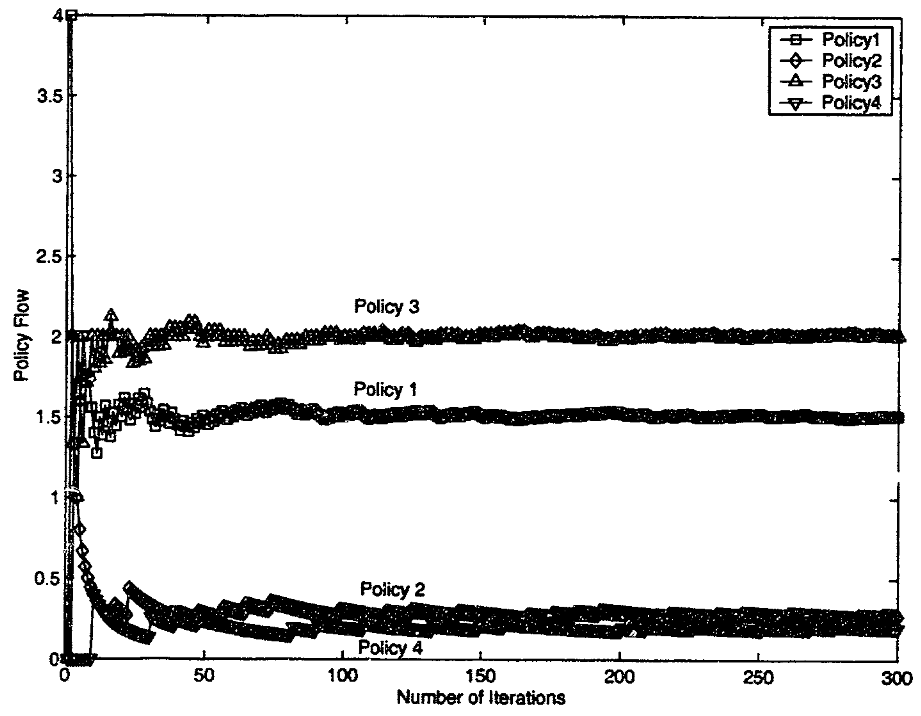


Figure 4-3: Convergence Results ($p = 0.1$)

We see that the policy flows tend towards the equilibrium solution very fast, after about 100 iterations. We see similar convergence results for other values of p. However this conclusion about the convergence rate cannot be generalized. Convergence rates may depend on specific applications.

4.4.4 Solution Discussion

We have already shown that the policy-based DTA model can be solved, with the desired outputs, including the policy flows and link travel time distributions. Another aim of the example is to show some features of a policy-based DTA model.

Path vs. Policy

Traditional DTA models work with paths. There are two ways of performing path-based traffic assignment when stochasticity exists. The first is the analog to the CE approximation presented in Chapter 3. One can take the mean value of any stochastic factor and thus transform the traffic network into a deterministic one. Then the deterministic path-based traffic assignment can be performed. In our example, this leads to the following. The volume-delay function of link a at time k can be replaced by an expected link performance function $C_a(k) = (1\times(1-p)^k+5\times(1-(1-p)^k))f_a(k)$. The first method is just an approximation, as there is no guarantee of any equilibrium conditions satisfied.

The second approach to path-based stochastic traffic assignment is to use our policy-based DTA model, but restrict the policies considered to paths only. The second is an exact model with the equilibrium condition that all used paths have the same and minimal expected travel times. In the following, we use "path-based" assignment to denote this method.

In our example, the two method happen to give the same path flows. This is due to the linear form of the volume-delay function, as the mean of a linear function of a random variable equals the linear function of the mean of the random variable. Generally we expect that the exact method will lead to less expected travel time than the approximation method.

We now study the difference between policy-based assignment and path-based assignment by comparing results from assigning flows to Policy 1 to 4 and from assigning flows only to Policy 1 and 2 (which are actually paths). See Table 4.4.4 for a summary of the equilibrium expected OD travel times obtained for the path-based assignment and the policy-based assignment.

We see that for $p \in (0,1)$, the policy-based assignment gives less expected OD travel times, compared to the path-based assignment. Furthermore, for $p \in (0,1)$, the relative difference between them suggests a decreasing function of p. This is intuitively correct. The information is useful only when an incident happens in time 1, for then we can conclude that the link has a reduced capacity in time 1. When

p	Policy-based	Path-based
0	4.00	4.00
0.1	5.30	5.62
0.2	6.25	6.59
0.3	6.97	7.24
0.4	7.48	7.68
0.5	7.88	8.00
0.6	8.16	8.22
0.7	8.35	8.38
0.8	8.47	8.49
0.9	8.54	8.55
1	8.57	8.57

Table 4.5: Equilibrium Expected OD Travel Times as Functions of p

no incident happens in time 1, the knowledge we have about time 1 is no more than the *a priori* one. Consequently if the incident is less likely, the fact of knowing its actual occurrence is more valuable. Thinking about it from another perspective, if the incident is very likely to happen, we already have the tendency to avoid it, and thus knowing its actual occurrence does not help us significantly.

For $p = 0$ or $p = 1$, the network changes to a deterministic one. In this case, all policies collapse to paths. Therefore it is intuitively correct that the policy-based and path-based assignment give the same results.

Collaboration can benefit everyone

We observe interesting phenomenon in the example results. For $p = 0.1$, we can see from Table 4.4.4, the equilibrium expected travel time is 5.30. If we assign flows only to true policies (i.e. Policy 3 and 4), we obtain an expected travel time of 5.21 for both policies. Under these flows, link a actually has a even lower expected travel time of 4.51. Therefore all users are actually better off under the non-equilibrium condition. However, as Policy 1 (link a) has less expected travel time, users will shift to Policy 1. The resulted equilibrium expected travel time, however, is higher. This situation is actually also present in the user optimal assignment in traditional deterministic assignment. This shows that if the users can collaborate, all of them can obtain less expected travel times.

4.5 Notation for a General Model

The physical traffic network is represented by a conceptual directed network as described in Subsection 2.2.1. In the following, the index g denotes a user group defined by risk taking behavior, the pair (r, s) denotes an O-D pair, the subscript μ denote a policy between (r, s), and K_{rs} is the set of policies between (r, s). We divide a link into two parts: the moving part and the queuing part. For a given link a, a_m denotes its moving part and a_q denotes its queuing part. t is the index for continuous time. All other notations are grouped into policy variables, link variables, link-policy variables, and time variables.

Policy variables:

$f_\mu^{rsg}(t)$	:	Departure flow rate on policy μ for user group g from origin r to destination s at time t
$f^{rsg}(t)$	:	Departure flow rate for user group g for O-D pair (r, s)

Link variables:

$U_{a_m}(t)(U_{a_q}(t))$	:	Cumulative entrance flow on the moving (queuing) part of link a during interval $[0, t]$
$u_{a_m}(t)(u_{a_q}(t))$	:	Entrance flow rate of the moving (queuing) part of link a at time t
$V_{a_m}(t)(V_{a_q}(t))$	:	Cumulative exit flow on the moving (queuing) part of link a during interval $[0, t]$
$v_{a_m}(t)(v_{a_q}(t))$	:	Exit flow rate of the moving (queuing) part of link a at time t
$X_{a_m}(t)(X_{a_q}(t))$	:	Load (Number of vehicles) of moving (queuing) part of link a at time t
$\tau_a(t)$	:	Travel time on link a for flows entering the link at time t
$\tau_{a_m}(t)(\tau_{a_q}(t))$	:	Travel time on the moving (queuing) part of link a for flows entering the moving (queuing) part at time t

L_a	:	Length of link a
$L_{a_q}(t)$	:	Queue length of link a at time t
$k_{a_m}(t)$	:	Density of the moving part of link a at time t
$k_{a,j}$	:	Jam density of link a
$w_{a_m}(t)$	:	Travel speed on the moving part of link a for flows entering the link at time t
$w_a^{min}(w_a^{max})$	:	minimum (free-flow) travel speed on link a
$X_a^c(t)$	:	Maximal allowable number of vehicles on link a at time t
$u_a^c(t)$	:	Maximal allowable inflow rate of link a at time t
$u_a^{max}(t)$	:	Upper-bound of the inflow rate of link a at time t
$v_a^c(t)$	:	Maximal allowable out-flow rate that can leave link a at time t

Link-policy flow variables:

$U_{a_m\mu}^{rsg}(t)(U_{a_q\mu}^{rsg}(t))$	:	Cumulative entrance flow of the moving (queuing) part of link a along policy p during interval $[0, t]$
$u_{a_m\mu}^{rsg}(t)(u_{a_q\mu}^{rsg}(t))$	:	Entrance flow rate of the moving (queuing) part of link a along policy μ at time t
$V_{a_m\mu}^{rsg}(t)(V_{a_q\mu}^{rsg}(t))$	:	Cumulative exit flow of the moving (queuing) part of link a along policy μ during interval $[0, t]$
$v_{a_m\mu}^{rsg}(t)(v_{a_q\mu}^{rsg}(t))$	:	Exit flow rate of the moving (queuing) part of link a along link pair μ at time t
$X_{a_m\mu}^{rsg}(t)(X_{a_q\mu}^{rsg}(t))$	:	Load of the moving (queuing) part of link a along policy μ at time t
$u_{a\mu}^{c,rsg}(t)$	:	Maximal allowable inflow rate of policy μ that can be accepted by link a at time t
$v_{a\mu}^{c,rsg}(t)$	:	Maximal allowable out-flow rate of policy μ that can leave link a at time t

Time Variables:

t	:	Index for continuous time
Δ	:	Minimum possible free flow link travel time over all links
δ	=	$\frac{\Delta}{M}$, M is the number of small interval with length δ within a Δ interval

4.6 Users' Policy Choice Model

Users of the traffic system are heterogeneous, and there are many ways to group users depending on applications. In the presence of stochasticity, the risk taking behavior is of importance. Therefore in this section, we introduce the risk taking behavior modeling and the corresponding "best" routing policy algorithms. We then divide travelers into three groups according their risk-taking behavior. We do not divide users according to their information access. Instead we assume that all users have access to the most updated online information provided by ATIS.

4.6.1 Risks in Routing Decision Making

In our earlier discussion of the routing model, the routing policies with minimal expected O-D travel times are found. In fact, when travelers are faced with uncertain travel times, they are concerned with not only the expected travel time of a routing choice, but also with the reliability of that choice. If one is sensitive to the loss a longer travel time would cause, one may prefer a routing policy with less variability. Similarly, if one is sensitive to the gain a shorter travel time would cause, he/she may prefer a routing policy with larger variability. Modeling risk taking behavior is an important part in routing decision making in practice. The following two subsections show how risk can be modeled under the framework of best routing policy problem. This involves a direct extension.

The extension is based on derivations in Mirchandani and Soroush [22], where disutility functions are used to model travelers' risk taking behavior. A disutility function $f(z)$ is a monotonically increasing function of travel time z. Assume travelers prefer a choice with less expected disutility, all other conditions of available choices equal. In this context, a linear disutility function models risk neutral behavior, and the best routing policy problem discussed in the previous two chapters can be viewed as a minimization problem over expected disutility functions with a special form of linear disutility function where $f(z) = z$.

An exponential disutility function, on the other hand, models risk taking behavior with constant risk averseness or proneness [19]. Assume the exponential disutility function is given by

$$g(z) = \alpha + \beta sgn(\gamma) exp(\gamma z) \tag{4.1}$$

where α, β, and γ are constants, such that β is positive and

$$sgn(\gamma) = \begin{cases} 1 & \text{if } \gamma > 0 \\ -1 & \text{if } \gamma < 0 \end{cases}$$

When γ is positive, the disutility function is convex and the travelers are risk averse. When γ is negative, the disutility function is concave and the travelers are risk prone [19]. α, β, and γ are to be calibrated. The best routing policy problem in a stochastic time-dependent network as stated in subsection 2.2.1 then can be extended as to find

$$\begin{aligned} \mu^* &= \arg\min_\mu \{E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[g(t_{x_S} - t_{x_0})]\} \\ &= \arg\min_\mu \{E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[\alpha + \beta sgn(\gamma) exp(\gamma(t_{x_S} - t_{x_0}))]\} \quad \text{for a given} \\ &= \arg\min_\mu \{E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[sgn(\gamma) exp(\gamma(t_{x_S} - t_{x_0}))]\} \end{aligned}$$

initial state x_0. We see that the constants α and β have no effect on the minimization of expected utility of a routing policy. We thus redefine the exponential disutility function as

$$f(t) = sgn(\gamma) exp(\gamma t) \tag{4.2}$$

without loss of generality, in the sense of obtaining the right optimal policy.

Define $d_\mu(x_0)$ as the expected disutility of routing policy μ from initial state x_0, i.e. $d_\mu(j,t,I) = E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[f(t_{x_S} - t_{x_0})]$. A recurrent expression of $d_\mu(x_0)$, which is convenient for stating optimality conditions, can be developed as follows.

$$\begin{aligned}
& d_\mu(x_0) \\
= \;& E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[sgn(\gamma)exp(\gamma(t_{x_S}-t_{x_0}))] \\
= \;& E_{\{x_0,x_1,\ldots,x_S\}\in M(x_0,\mu)}[sgn(\gamma)exp(\gamma(t_{x_1}-t_{x_0}))exp(\gamma(t_{x_S}-t_{x_1}))] \\
= \;& E_{x_1|\mu(x_0)\in x_1}[exp(\gamma(t_{x_1}-t_{x_0}))E_{\{x_1,\ldots,x_S\}\in M(x_1,\mu)}[sgn(\gamma)exp(\gamma(t_{x_S}-t_{x_1}))]] \\
= \;& E_{x_1|\mu(x_0)\in x_1}[exp(\gamma(t_{x_1}-t_{x_0}))d_\mu(x_1)]
\end{aligned}$$

Define $x_0 = \{j,t,I\}$ and $x_1 = \{k, t+C_{jk,t}, I'\}$, where $k=\mu(x_0)$, then the above expression of $d_\mu(x_0)$ can be written as

$$\begin{aligned}
d_\mu(j,t,I) &= E_{C_{jk,t},I'}[exp(\gamma C_{jk,t})d_\mu(k,t+C_{jk,t},I')|I] \\
&= E_{C_{jk,t}}[exp(\gamma C_{jk,t})E_{I'}[d_\mu(k,t+C_{jk,t},I')]|I]
\end{aligned}$$

Therefore $\forall j \in N\backslash\{d\}, \forall t \in T, \forall I$ that is possible at node j and at time t, $d_{\mu^*}(x)$ and μ^* are solutions of the following system of equations:

$$d_{\mu^*}(j,t,I) = \min_{k\in A(j)}\{E_{C_{jk,t}}[exp(\gamma C_{jk,t})E_{I'}[d_{\mu^*}(k,t+C_{jk,t},I')]|I]\} \tag{4.3}$$

$$\mu^*(j,t,I) = \arg\min_{k\in A(j)}\{E_{C_{jk,t}}[exp(\gamma C_{jk,t})E_{I'}[d_{\mu^*}(k,t+C_{jk,t},I')]|I]\} \tag{4.4}$$

with the boundary conditions: $d_{\mu^*}(d,t,I) = sgn(\gamma), \mu^*(d,t,I) = d, \forall t \in T, \forall I$ that is possible at node d and at time t.

We note that the optimality conditions for the BRP problem aiming at minimizing expected travel time and that aiming at minimizing expected exponential disutility function have the same structure. Therefore algorithms developed for BRP variants that minimize expected travel times could be used, with minor changes, to determine solutions minimizing exponential disutility functions incorporating risk behavior. Note that $d_{\mu^*}(j,t,I)$ is derived based on the definition of disutility function as in Equation 4.2. The true expected disutility as defined in Equation 4.1 can be obtained by applying a linear transformation: $\alpha + \beta d_{\mu^*}(j,t,I)$.

4.6.2 Classification of Users

We assume that ATIS provides perfect online information (see definition in Subsection 2.2.2) and all users have access to the global network conditions through communications with the ATIS. Users could have perception errors in obtaining the information from ATIS. For the sake of ease in presenting the model, we assume that there are no perception errors. This is to say, that all users have perfect online information. As we discussed before, even with the same information, the same time and the same current node, users with different risk-taking behavior will make different routing decisions. Therefore we classify users into three types accordingly:

type 1: users who are risk neutral

Users who are risk neutral have disutility functions $g(z) = z$, i.e. they consider only the expected value of O-D travel time, not the variability. Therefore they will follow the routing policy that minimizes the expected travel time to the destination node. Specifically, routing policies generated by Algorithm DOT-SPI from the routing model will be followed by this type of users.

type 2: users who are risk averse

Users who are risk averse have disutility functions $g(z) = \alpha + \beta exp(\gamma z)$. Therefore they will follow the routing policy that minimizes the expected disutility as defined to the destination node. The optimality conditions presented in Equation (4.4) can be used to obtain the routing policy that minimizes the expected disutility to the destination node.

type 3: users who are risk prone

The processing of users who are risk prone are conceptually the same as that for the users who are risk averse. The only difference lies on the sign of the constant γ.

4.7 The Dynamic Network Loading Model

In this section, we develop a flow-based policy-based dynamic network loading model. This work is built upon previous works by Chabini and He [11] and Chabini and Lan [12]. The common feature of these works is that the network loading model is expressed through a system of flow equations. Specifically the work by Chabini and

Lan [12] considers the spill-back of queues which is a desirable feature in modeling congested networks. The modeling of queues is done by enforcing the limit on maximal inflow rate, maximal out-flow rate and storage capacity of any link. We will take the same approach to model queues.

Numerous random factors in the supply side can be captured as a change in capacities. For example, an incident generally blocks several lanes, and thus reduces the storage capacity of the link. Red lights can be viewed as a device to reduce the maximal outflow rate to zero, while green lights recover the maximal outflow rate to the normal value. The model presented in this section is not claimed to be the best network loading model in terms of modeling traffic phenomena. It is adopted instead as it is relatively realistic and suffices to illustrate the implications of policy-based network loading.

4.7.1 Travel Times on Links with Queue and Varied Out-flow Rates

We divide a link into two parts in the presence of queue: a moving part and a queuing part. Assume that vehicles in a queuing part move in jam density. See Figure 4-4.

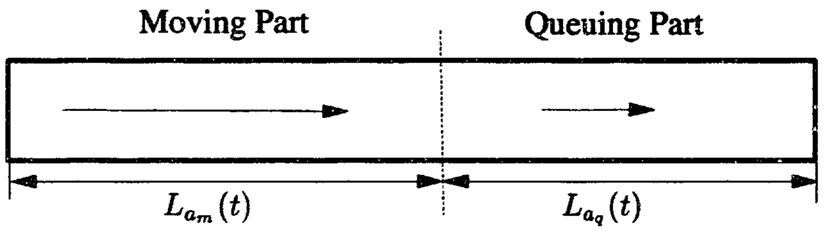


Figure 4-4: Moving Part and Queuing Part of a Link

We also make the following two assumptions about link travel times:

1. link travel times are bounded from below by a positive number, and

2. the travel time of a link depends only on the current and/or past traffic conditions on the link.

These two assumptions are realistic because (1) a link has a minimum length and the travel speed is finite and is no greater than the free flow travel time of that link and

(2) the travel time for a user entering the link usually depends only on the number of vehicles that entered the link earlier.

The travel speed on the moving part $w_{a_m}(t)$ can be determined by the modified Greenshields' model as follows:

$$w_{a_m}(t) = w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{k_{a_m}(t)}{k_{aj}})^{\alpha}]^{\beta} \tag{4.5}$$

where $k_{a_m}(t) = \frac{X_{a_m}(t)}{L_a - L_{a_q}(t)}$, and the parameters α and β need to be calibrated for each link.

The travel time on the moving part, $\tau_{a_m}(t)$ can be determined approximated as the follows:

$$\tau_{a_m}(t) = \frac{L_a \times k_{aj} - X_{a_q}(t)}{w_{a_m}(t) \times k_{aj} + (u_{a_q}(t) - v_{a_q}(t))} \tag{4.6}$$

If the FIFO condition is satisfied on queuing part (one cannot possibly pass others in a queue), all flows enter the queuing part at time interval $[0, t]$ will exit the queuing part at time interval $[0, t + \tau_{a_q}(t)]$. If we assume FIFO is satisfied for the queuing part, the travel time on the queuing part can determined by the following equation of $\tau_{a_q}(t)$, assuming cumulative inflows and outflows have been determined:

$$U_{a_q}(t) = V_{a_q}(t + \tau_{a_q}(t)) \tag{4.7}$$

We do not actually enforce FIFO in the queuing part in the formulation. The above equation can be used as an approximation to determine queuing travel times.

4.7.2 A Moving-Queuing Model for Policy-Based Dynamic Network Loading

We develop a moving-queuing model for the policy-based dynamic network loading, based on the work by Chabini and Lan [12]. Basically we assume that the queue can only occur starting from the tail of a link, and it can propagate to the head of the link. A general depiction of a link with the flow variables are shown in Figure 4-5. This

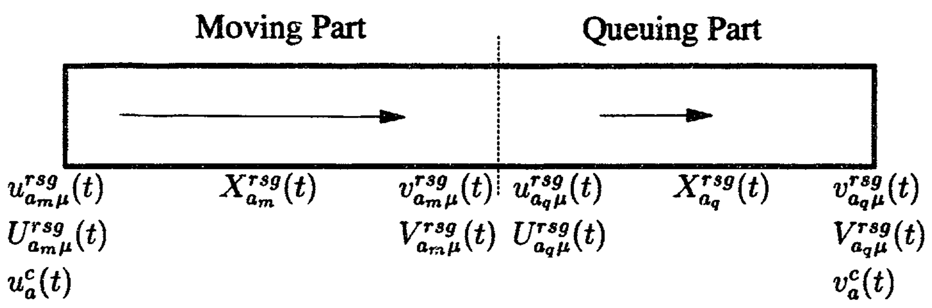


Figure 4-5: Moving-Queuing Model with Variables

model is suitable for queues with fixed front end. It models adequately the formation of queues, because generally the formation of a queue is due to a bottleneck with reduced capacity which is the front end of the queue. The bottleneck is generally fixed, for example, an incident, the narrowing of a road, a red light, etc. However, this moving-queuing model cannot model the dissipation of a queue adequately, where both the front end and the back end of the queue are moving. Chabini and Lan [12] has also proposed a moving-queuing-moving model which can consider the dissipation of queues. In our DTA model, only the moving-queuing model is adapted to show the idea, but the adaptation of the moving-queuing-moving model can be done similarly.

Augmentation of the Network and Spillback of Queues at Origin Nodes

Before presenting the network loading formulations, we augment the network to consider the case where the spillback of queues reaches the origins. For each origin node r, a virtual origin r' and a virtual link (r', r) is added to the network. Link (r', r) has infinite maximal inflow rate, infinite maximal out-flow rate, and infinite storage capacity. It could have a moving part and a queuing part as a regular link, to model the spillback to the origin. However, travel times on both parts are always zero, i.e. Equation 4.6 and Equation 4.7 are not applicable here.

Queuing States of Link a and Adjacent Link Pair (a', a)

Link a could be in three states at time t as shown in Figure 4-6.

Link a is in state 1 if the partial load of queue, $X_{a_q}(t)$, is equal to zero. Link a is in state 2 if $X_{a_q}(t) = X^c_a(t)$. Link a is in state 3 if $0 < X_{a_q}(t) < X^c_a(t)$.

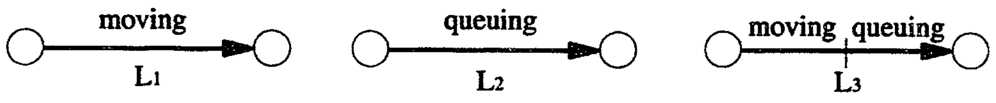


Figure 4-6: States of a Link

Accordingly, adjacent link pair (a', a) could be in four states at time t in terms of queuing status of the downstream end of link a' and the upstream end of a. The four states are depicted in Figure 4-7.

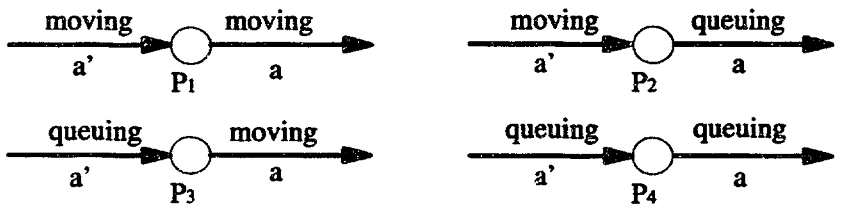


Figure 4-7: Status of a Link Pair

The formulation of the dynamic network loading (DNL) problem can be different for different link queuing states and link pair queuing states. In the rest of the subsection, we present a dynamic network loading model for all the cases. Most of the differences are trivial, with only changes in notations. We use L_1, L_2, L_3 to denote the three queuing states for a given link, and P_1, P_2, P_3, P_4 to denote the four queuing states of a given adjacent link pair.

Determining the Next Link of Policy μ at time t

A routing policy is a decision rule based on on-line information. It maps link travel time realizations to routing decisions, i.e. the next node (link) to take. Therefore one cannot know which link he/she should enter after traversing a until he/she is at the end of that link. As we have assumed, the input routing policy is based on the assumption of perfect online information, i.e. the users know all link travel time realizations up to the current time k. To determine the next link of policy μ at time t, we must translate the current available link travel times $\tau_a(t'), \forall t' < t$ to an event set at time t in the definition of policy μ. However as we have pointed out in the discussion of Equation 4.7, if queues exist, we are not able to obtain all the link travel times up to time t. Furthermore, even if we have all the link travel time realizations, they might not be exactly the same as any event set of the routing policy μ at time t. We solve this problem by using a weighted least-square method. First we approximate all unavailable link travel time realizations up to time t with their latest available realizations respectively: $\tau_a(t') = \tau_a(t_0), \forall t_0 < t' \leq t$, where t_0 is the largest time index with known queuing travel time for link a. Next we find an event set closest to the approximated link travel time realizations, in terms that the weighted second-order norm between the approximated link travel time realizations and the event set is the smallest among all possible event sets at time t. The weights can be customized, and intuitively link travel time realizations of a larger time index have larger weight than those with smaller time indexes. We denote by $\mu(a, t)$ the next link of link a along policy μ at time t. The link travel time realizations $\tau_a(t'), \forall t' < t$ are omitted, since they are unique during a single loading process.

Modeling the Moving Part of Link a

The DNL model for the moving part of link a is formulated as the following system of equations. All link states are per to link a, and all adjacent link pair states are per to link pair (a', a).

Link Dynamics Equations

$$\frac{dX^{rsg}_{a_m\mu}(t)}{dt} = u^{rsg}_{a_m\mu}(t) - v^{rsg}_{a_m\mu}(t) \qquad \forall (r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.8}$$

Flow Conservation Equations

If link a is not a virtual link:

P_1 (moving-moving):

$$u^{rsg}_{a_m\mu}(t) = v^{rsg}_{a'_m\mu}(t) \qquad \forall (r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a', t), \forall a \tag{4.9}$$

P_3 (queuing-moving):

$$u^{rsg}_{a_m\mu}(t) = v^{rsg}_{a'_q\mu}(t) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a,t), \forall a \tag{4.10}$$

If link a is a virtual link:

$$u^{sg}_{a_m\mu}(t) = f^{rsg}_{\mu}(t) \qquad \forall(r,s), \forall g, \forall a = (r', r) \forall \mu \in K_{rs} \tag{4.11}$$

Flow Propagation Equations

$$V^{rsg}_{a_m\mu}(t) = \int_{w \in \{z | z + \tau_{a_m}(z) \le t\}} u^{rsg}_{a_m\mu}(w) dw \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.12}$$

Initial Conditions

$$U^{sg}_{a_m\mu}(0) = 0, V^{sg}_{a_m\mu}(0) = 0, X^{rsg}_{a_m\mu} = 0, \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.13}$$

Modeling the Queuing Part of Link a

The hard part of modeling the queuing part lies in the computation of travel time on queuing part $\tau_a(t)$. It can be obtained by solving Equation 4.7. However, the solution depends on flow variable values at future times and thus cannot be obtained when vehicles just join the queue. Without queuing travel times, the out-flow rate then cannot be determined using flow propagation equations similar to Equation 4.12. We resort to the maximal inflow rate and out-flow rate to determine out-flow rate for the queuing part.

The outflow rate of any link is constrained by $v^c_a(t)$, the maximal allowable outflow rate of the link and $u^c_a(t)$, the maximal allowable inflow rate of the next link. We assume $v^c_a(t)$ is mainly determined by infrastructure profile and is given for a single loading process. For example, if an incident occurs from 9am to 11am and blocks half the lanes, $v^c_a(t)$ can be cut in half for 9am $\le t \le$ 11am. The determination of

u_a^c is more involved, as it depends not only on infrastructure profile, i.e. the upper-bound of inflow rate u_a^{max}, but also on the queuing states of link a. If link a is all queuing, i.e. the storage capacity of link a is reached, and if the speed of wave propagation is infinite, the number of vehicles that can be accepted by link a during a time interval is at most the number of vehicles that exit link a during that time interval. Mathematically speaking, we have:

$$u_a^c(t) = \begin{cases} u_a^{max}(t), & \text{if } a \text{ is in } L_1 \text{ or } L_3 \\ \min(u_a^{max}(t), v_{a_q}(t)), & \text{if } a \text{ is in } L_2 \end{cases} \quad \forall a \tag{4.14}$$

Note that by assuming the speed of wave propagation is infinite, the queue is like a train: whenever the head of the train moves, the tail of the train moves at the same time. This is an approximation to the realistic situation.

We then have to allocate the maximal allowable outflow rate and maximal allowable inflow rate to users in different groups with different O-D pairs and routing policies. The allocation is based on flow ratios:

$$u_{a\mu}^{c,rsg}(t) = \begin{cases} u_a^c(t) \times f_1(u_{a_m\mu}^{rsg}(t), u_{a_m}(t)), & \text{if } a \text{ is in } L_1 \text{ or } L_3 \\ u_a^c(t) \times f_1(u_{a_q\mu}^{rsg}(t), u_{a_m}(t)), & \text{if } a \text{ is in } L_2 \end{cases} \tag{4.15}$$
$$\forall (r,s), \forall g, \forall \mu \in K_{rs}, \forall a$$

$$v_{a\mu}^{c,sg}(t) = \begin{cases} v_a^c(t) \times h_1(v_{a_m\mu}^{rsg}(t), v_{a_m}(t)), & \text{if } a \text{ is in } L_1 \\ v_a^c(t) \times h_1(v_{a_q\mu}^{rsg}(t), v_{a_m}(t)), & \text{if } a \text{ is in } L_2 \text{ or } L_3 \end{cases} \tag{4.16}$$
$$\forall (r,s), \forall g, \forall \mu \in K_{rs}, \forall a$$

Functions $f_1(.), f_2(.), h_1(.), h_2(.)$ can be defined by users.

The formulation of the DNL problem for the queuing part is expressed by the following system of equations:

<u>Link Dynamics Equations</u>

$$\frac{dX_{a_q\mu}^{rsg}(t)}{dt} = u_{a_q\mu}^{rsg}(t) - v_{a_q\mu}^{rsg}(t) \qquad \forall (r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.17}$$

Flow Conservation Equations

L_2 (queuing) and P_2 (moving-queuing):

$$u^{rsg}_{a_q\mu}(t) = v^{rsg}_{a'_m\mu}(t) \qquad \forall(r,s), \forall g, \forall\mu \in K_{rs}, a = \mu(a',k), \forall a \tag{4.18}$$

L_2 (queuing) and P_4 (queuing-queuing):

$$u^{rsg}_{a_q\mu}(t) = v^{rsg}_{a'_q\mu}(t) \qquad \forall(r,s), \forall g, \forall\mu \in K_{rs}, a = \mu(a',k), \forall a \tag{4.19}$$

L_3 (moving-queuing):

$$u^{rsg}_{a_q\mu}(t) = v^{rsg}_{a_m\mu}(t) \qquad \forall(r,s), \forall g, \forall\mu \in K_{rs}, \forall a \tag{4.20}$$

Flow Propagation Equations

If link a is not the last link for destination s:

$$v^{rsg}_{a_q\mu}(t) = \min(v^{c,rsg}_{a\mu}(t), u^{c,rsg}_{a''\mu}(t)) \qquad \forall(r,s), \forall g, \forall\mu \in K_{rs}, a'' = \mu(a,t), \forall a \tag{4.21}$$

If link a is the last link for destination s:

$$v^{rsg}_{a_q\mu}(t) = v^{c,rsg}_{a\mu}(t) \qquad \forall(r,s), \forall g, \forall\mu \in K_{rs}, \forall a \tag{4.22}$$

Initial Conditions

$$U^{rsg}_{a_q\mu}(0) = 0, V^{rsg}_{a_q\mu}(0) = 0, X^{rsg}_{a_q\mu}(0) = 0, \forall(r,s), \forall g, \forall\mu \in K_{rs}, \forall a \tag{4.23}$$

4.8 Solution Algorithms

We descretize the continuous time into small intervals. For each interval k, the continuous variables are assumed to be constants within that interval. These variables include the O-D trips, various variables in the dynamic network loading problem, and the link travel times in a STD network as an input to the BRP problem.

4.8.1 User's Policy Choice Algorithm

The users' policy choice algorithm takes as input the the dynamic O-D trips and the best routing policies for all O-D pairs and all user groups. It produces policy flows $f_{\mu}^{rsg}, \forall(r,s), \forall g, \forall \mu \in K_{rs}$. We assign all flows to their corresponding best routing policies.

$$f_{\mu}^{rsg}(t) = \begin{cases} f^{rsg}(t), & \text{if policy } \mu \text{ is a best routing policy for user group } g, \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

In case that more than one policies are optimal, we can assign users equally to these optimal policies. Denote by $N_{min}(k)$ the number of optimal policies for user group g of O-D pair (r,s) at time k. Therefore $f_{\mu}^{rsg}(k)$ is given by:

$$f_{\mu}^{rsg}(t) = \begin{cases} f^{rsg}(t)/N_{min}(k), & \text{if policy } \mu \text{ is a best routing policy for user group } g, \\ 0, & \text{otherwise.} \end{cases} \tag{4.25}$$

The method of the policy choice model is as follows: *for all O-D pair* $(r,s), \mu \in K_{rs}, g \in \{1,2,3\}$, *compute* f_{μ}^{rsg} *using (4.24) or (4.25).*

4.8.2 Algorithm of Dynamic Network Loading Model

In this subsection, we present a solution algorithm for the policy-based dynamic network loading model. The input to this model is the policy flows for each O-D pair and each user group f_{μ}^{rsg}), and the stochastic and dynamic supply. The output of the model is the sample distribution of link travel times of all links at all times. The algorithm is an adaptation of C-load algorithm developed by Chabini and He [11] which was used to solve a path-based DNL problem with no queues.

The development of the solution algorithm to the dynamic network loading model is based on a discretized version of the system of equations. Denote Δ as the minimum link travel time over all links and all realizations. We choose the interval length $\delta = \Delta/M$, where M is a positive integer. Each interval is indexed by an integer k, and the k^{th} interval represents $[k\delta, (k+1)\delta)$. For each interval k, the continuous variables are assumed to be constants within that interval. We present a discretized DNL model as follows:

Moving Part of Link a

Link Dynamics Equations

$$X^{rsg}_{a_m\mu}(k) = U^{rsg}_{a_m\mu}(k) - V^{rsg}_{a_m\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.26}$$

Flow Conservation Equations

If link a is not a virtual link:

P_1 (moving-moving):

$$u^{rsg}_{a_m\mu}(k) = v^{rsg}_{a'_m\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a',t), \forall a \tag{4.27}$$

P_3 (queuing-moving):

$$u^{rsg}_{a_m\mu}(k) = v^{rsg}_{a'_q\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a,t), \forall a \tag{4.28}$$

If link a is a virtual link:

$$u^{sg}_{a_m\mu}(k) = f^{rsg}_{\mu}(k) \qquad \forall(r,s), \forall g, \forall a = (r',r) \forall \mu \in K_{rs} \tag{4.29}$$

Flow Propagation Equations

$$V^{rsg}_{a_m\mu}(k) = \sum_{j \in \{j: 0 \le j\delta + \tau_{a_m}(j) \le k\delta\}} u^{rsg}_{a_m\mu}(k)\delta \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.30}$$

$v^{rsg}_{a_m\mu}(k)$ can be calculated approximately as follows:

$$v^{rsg}_{a_m\mu}(k) = \frac{V^{rsg}_{a_m\mu}(k) - V^{rsg}_{a_m\mu}(k-1)}{\delta} \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.31}$$

Initial Conditions

$$U^{rsg}_{a_m\mu}(0) = 0, V^{rsg}_{a_m\mu}(0) = 0, X^{rsg}_{a_m\mu}(0) = 0 \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.32}$$

$U^{rsg}_{a_m\mu}(k)$ can be calculated as follows:

$$U^{rsg}_{a_m\mu}(t) = \sum_{j=0}^{k-1} u^{rsg}_{a_m\mu}(j)\delta \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.33}$$

Queuing Part of Link a

Link Dynamics Equations

$$X^{rsg}_{a_q\mu}(k) = U^{rsg}_{a_q\mu}(k) - V^{rsg}_{a_q\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K, \forall a \tag{4.34}$$

Flow Conservation Equations

L_2 (queuing) and P_2 (moving-queuing):

$$u^{rsg}_{a_q\mu}(k) = v^{rsg}_{a'_m\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a',k), \forall a \tag{4.35}$$

L_2 (queuing) and P_4 (queuing-queuing):

$$u^{rsg}_{a_q\mu}(k) = v^{rsg}_{a'_q\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a = \mu(a',k), \forall a \tag{4.36}$$

L_3 (moving-queuing):

$$u^{rsg}_{a_q\mu}(k) = v^{rsg}_{a_m\mu}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.37}$$

Flow Propagation Equations

If link a is not the last link for destination s:

$$v_{a_q\mu}^{rsg}(k) = \min(v_{a\mu}^{c,rsg}(k), u_{a''\mu}^{c,rsg}(t)) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, a'' = \mu(a,t), \forall a \quad (4.38)$$

If link a is the last link for destination s:

$$v_{a_q\mu}^{rsg}(k) = v_{a\mu}^{c,rsg}(k) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \quad (4.39)$$

Please refer to Equation 4.41 through Equation 4.43 for the formula of determining $v_{a\mu}^{c,rsg}(k), u_{a\mu}^{c,rsg}(k)$, and $v_{a}^{c,rsg}(k)$.

Initial Conditions

$$U_{a_q\mu}^{rsg}(0) = 0, V_{a_q\mu}^{rsg}(0) = 0, X_{a_q\mu}^{rsg}(0) = 0, \forall(r,s), \forall g, \forall p \in K_{rs}, \forall a \quad (4.40)$$

The allocation of capacities for time interval k is straightforward at the first look. However, we do not know the flow values at time k before determining the capacities at time k. Thus we have to approximate the flow values at time k by their values at time $k-1$. Specifically we have the following:

$$u_a^c(k) = \begin{cases} u_a^{max}(k), & \text{if } a \text{ is in } L_1 \text{ or } L_3 \\ \min(u_a^{max}(k), v_{a_q}(k-1)), & \text{if } a \text{ is in } L_2 \end{cases} \quad \forall a \qquad (4.41)$$

$$u_{a\mu}^{c,sg}(k) = \begin{cases} u_a^c(k) \times f_1(u_{a_m\mu}^{rsg}(k-1), u_{a_m}(k-1)), & \text{if } a \text{ is in } L_1 \text{ or } L_3 \\ u_a^c(k) \times f_2(u_{a_m\mu}^{rsg}(k-1), u_{a_m}(k-1)), & \text{if } a \text{ is in } L_2 \end{cases} \qquad (4.42)$$
$$\forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a$$

$$v_{a\mu}^{c,sg}(k) = \begin{cases} v_a^c(k) \times h_1(v_{a_m\mu}^{rsg}(k-1), v_{a_m}(k-1)), & \text{if } a \text{ is in } L_1 \\ v_a^c(k) \times h_2(v_{a_m\mu}^{rsg}(k-1), v_{a_m}(k-1)), & \text{if } a \text{ is in } L_2 \text{ or } L_3 \end{cases} \tag{4.43}$$
$$\forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a$$

$U_{a_q\mu}^{rsg}(k)$ can be calculated as follows:

$$U_{a_q\mu}^{rsg}(k) = \sum_{j=0}^{k-1} u_{a_q\mu}^{rsg}(j) \qquad \forall(r,s), \forall g, \forall \mu \in K_{rs}, \forall a \tag{4.44}$$

$V_{a_q\mu}^{rsg}(k)$ can be calculated as follows:

$$V_{a_q\mu}^{rsg}(k) = V_{a_q\mu}^{rsg}(k-1) + v_{a_q\mu}^{rsg}(k)\delta \tag{4.45}$$

Travel Times on Moving and Queuing Part

$$w_{a_m}(k) = w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{k_{a_m}(k)}{k_{aj}})^{\alpha}]^{\beta}$$

$$\tau_{a_m}(k) = \frac{L_a \times k_{aj} - X_{a_q}(k)}{w_{a_m}(k) \times k_{aj} + (u_{a_q}(k) - v_{a_q}(k))} \tag{4.46}$$

$$U_{a_q}(k) = V_{a_q}(k + \tau_{a_q}(k)) \tag{4.47}$$

It is obvious that we cannot solve for $\tau_{a_q}(k)$ at time interval k. However we have a procedure to determine $\tau_{a_q}(j), j < k$ as follows:

Procedure_Queuing_Time(a, k): Searching τ_{a_q} at Time k

1. Let k' be the largest time index with known τ_{a_q}

2. $k'' = k' + 1$

3. If $U_{a_q}(k'') = V_{a_q}(k)$, then $k' = k' + 1, \tau_{a_q}(k'') = k - k''$

Total travel time on link a if entering the link at time interval k:

$$\tau_a(k) = \tau_{a_m}(k) + \tau_{a_q}(k + \lfloor \frac{\tau_{a_m}(k)}{\delta} \rfloor) \tag{4.48}$$

We use the following conditions to determine the status of link a at time k:

Procedure_Link_State(a, k): Determining the State of Link a at Time k

1. Approximate $X_{a_q}(k)$ by setting it equal to $X_{a_q}(k-1) + (u_{a_q}(k-1) - v_{a_q}(k-1))\delta$
2. If link a is in status L_1 at time $k-1$, then go to Step 6
3. If $X_{a_q}(k) \leq 0$, then L_1
4. If $X_{a_q}(k) \geq X_a^c(k)$, then L_3
5. If $0 < X_{a_q}(k) < X_a^c(k)$, then L_2
6. If $v_{a_m\mu}^{rsg}(k) > \min(v_{a\mu}^{c,rsg}(k), u_{a''\mu}^{c,rsg}(k)), \exists(r,s), \exists g, \exists \mu \in K_{rs}$, then L_3, else L_1

We use an algorithm adapted from C-load [11] [12] to solve the DNL problem in a chronological order. The DNL problem takes as input the policy flows $f_\mu^{rsg}, \forall(r,s), \forall g, \forall \mu \in K_{rs}$, and produces link travel time sample distributions with a pre-specified sample size R. The statement of the algorithm is as follows:

C-Load for the Policy-Based Dynamic Network Loading Model

Step 0 (Initialization)

0.1 Determine Δ by $\Delta = \min_a \frac{L_a}{w_a^{max}}$

0.2 Determine $M = \Delta/\delta$

0.3 R: the number of joint realizations of link travel times

0.4 $i = 0$ (the counter for number of realizations)

0.5 $j = 0$ (the counter for time indexes)

Step 1 (Loading)

1.1 Sample from stochastic dynamic supplies
to obtain $v_a^c(k), u_a^{max}(k), \forall a, \forall k$

```
1.2 (A Single Process of Loading)
    1.2.1 Initialization (4.32)(4.40)
    1.2.2 (Loading of the $j^{th}$ $\Delta$ interval. See "1.2.2 of C-Load")
    1.2.3 If network is empty, go to Step 2
          otherwise $j = j + 1$, go to Step 1.2.2
1.3 (Stopping Criterion)
    $c^i_{jk,t} = \tau_a(t)$, $p_r = 1/R$
    If $i = R$, STOP
    Otherwise $i = i + 1$, and go to Step 1.1
```

Step 1.2.2 of C-Load

```
1.3.2 For $k = jM$ to $(j+1)M - 1$ do:
    1.3.2.1 Determine $a'' = \mu(a,k), \forall a$
    1.3.2.2 For $a \in A, \mu \in K_{rs}, g$, do:
        Compute $V^{rsg}_{a_m\mu}(k)$ (4.30)
        Compute $v^{rsg}_{a_m\mu}(k)$ (4.31)
        Determine the state of link $a$ by calling Procedure_Link_State($a, k$)
        If $L_1$: do nothing
        If $L_2$ or $L_3$: Compute $v^{rsg}_{a_q\mu}(k)$ (4.38)(4.39)
                           Compute $V^{rsg}_{a_q\mu}(k)$ (4.45)
    1.3.2.3 For $a \in A, \mu \in K_{rs}, g$, do:
        Determine the state of $(a', a)$ by calling Procedure_Link_State($a', k$)
        For moving part:
                Compute $u^{rsg}_{a_m\mu}$ (4.27)(4.28)(4.29)
                Compute $U^{rsg}_{a_m\mu}$ (4.33)
                Compute $X^{rsg}_{a_m\mu}$ (4.26)
        For queuing part:
                Compute $u^{rsg}_{a_q\mu}$ (4.35)(4.36)(4.37)
                Compute $U^{rsg}_{a_q\mu}$ (4.44)
```

```
            Compute $X^{rsg}_{a_q\mu}$ (4.34)
    1.3.2.4 Compute $\tau_{a_m}(k)$ (4.46)
        Compute $\tau_{a_q}(j), j < k$ by calling Procedure_Queuing_Time($a, k$)
    1.3.2.4 Compute $u^{c,rsg}_{a\mu}(k)$ and $v^{c,rsg}_{a\mu}(k)$ (4.41)(4.42)(4.43)
```

4.8.3 The Stochastic Policy-based DTA Heuristic

The idea of the solution algorithm is to find a solution to the policy-based DTA model by an iterative process on path flows. At each iteration, the policy flows are updated by combining the results from the current iteration and the previous iteration. We use MSA [27] to update policy flows. Since no proof of convergence is available at this moment, the method is heuristic for the DTA problem. The algorithm is presented as follows:

Policy-Based Stochastic DTA Heuristic

```
Step 0 (Initialization)
    0.1: $N$ = maximal number of iterations;
    0.2: $R$ = number of realizations of link travel times
    0.3: Augment the network by adding virtual nodes and links for origins
         (See Subsection 4.7.2);
    0.4: Compute initial policy flows $\{f^{rsg^{(0)}}_{\mu}(k)\}$
         from free-flow link travel times (NOTE: a path is a policy);
    0.5: $n = 0$ (the counter of iteration);
Step 1 (Main Loop)
    1.1: Run the dynamic network loading model using C-load
         to obtain $\{c^r_{jk,t}, r = 1, ..., R\}$
    1.2: Run the BRP algorithm to obtain optimal routing policies
         for all $(r, s)$, all $g$, based on $\{c^r_{jk,t}, r = 1, ..., R\}$;
    1.3: Compute $f^{rsg}_{\mu}(k)$ by the policy choice algorithm;
    1.4: Update policy flows:
```

$f_\mu^{rsg^{(n+1)}}(k) = f_\mu^{rsg^{(n)}}(k) + \alpha^{(n)}[f_\mu^{rsg}(k) - f_\mu^{rsg^{(n)}}(k)]$,

where $\alpha^{(n)} = 1/(n+1), g = 1, 2, 3$.

Step 2 (Stopping Criterion)

If $n = N$, STOP

Otherwise, $n = n + 1$, and go to Step 1

4.9 Concluding Remarks

In this chapter, we studied the policy-based stochastic dynamic traffic assignment. We gave a conceptual framework for this model, composed of three models: the routing policy generation model, the users' policy choice model, and the dynamic network loading model. We have already studied the routing policy generation model in Chapter 2 and Chapter 3. We gave the policy-based stochastic DTA equilibrium condition. An illustrative example is given to show the process of policy-based assignment and study the characteristics of a policy-based assignment. We then studied the users' policy choice model. We modeled the users' risk taking behavior by using disutility functions and showed that a straighforward extension of the BRP algorithm can be used to solve the minimum expected disutility problem. We then classified users into three groups based on the risk taking behavior. The dynamic network loading model is formulated based on the work by Chabini and Lan [12]. Solution algorithms are then presented.

Chapter 5

Conclusions and Future Directions

We studied the best routing policy problems in stochastic time-dependent networks. This problem is a fundamental research problem with a wider application domain in traffic. This includes traffic networks where this problem arises in the development of dynamic traffic assignment methods. There are many variants of the BRP problem in STD networks, however they can be integrated in a framework. We established such a framework, including a general description of the STD network, the decision process, the problem statement, and the optimality conditions. We provided a comprehensive taxonomy of the BRP problem, based on network statistical dependency and information access. These two factors determine the current-information based on which the routing decisions are made. Numerous variants exist according to the taxonomy, and we provided insights into most of them, focusing on the specification of current-information. We then studied in details a variant (termed the POI variant) which is needed in dynamic traffic assignment models. The POI variant takes into account the statistical dependency among link travel times and the role of information in routing decision making, which is a realistic depiction of traffic systems equipped with ATIS and/or ATMS. An exact algorithm (Algorithm DOT-SPI) was designed and implemented for this variant.

The complexity analysis of Algorithm DOT-SPI revealed the need to design good approximations for the BRP problem. Four approximations were presented. Their properties were studied both theoretically and computationally. There is a trade-off between effectiveness and efficiency for all approximations, i.e. they could have satisfactory running times, but their results could be arbitrarily worse in absolute value than those obtained by running the exact algorithm. The computational tests studied the relationship between some parameters and the performance of approximations.

We studied for the first time a policy-based dynamic traffic assignment model. This DTA model outputs sample distribution of link travel times and other measures of effectiveness in interest. We proposed a conceptual framework and stated the equilibrium condition for the policy-based user optimal dynamic traffic assignment. We revealed the difference between policy-based and traditional path-based DTA models through an illustrative example. We developed a users' policy choice model and a dynamic network loading model. Solution algorithms were designed for both models. A DTA heuristic was developed based on the work on the routing model and the users' policy choice model and dynamic network loading model.

Future research in the BRP problem can be in the following directions:

1. Identify the variants with realistic assumptions on network statistical dependency and information access that are suitable for traffic applications. Intuitively local information access and partial statistical dependency is the correct choice, but further research work is required to obtain the specific form that trades off realism and model tractability.

2. The mechanism to deploy the BRP algorithms and approximations in actual traffic applications. For example, how to obtain the joint realizations of link travel times needed for Algorithm DOT-SPI? What if the observed link travel time realizations do not comply with the a priori distributions? For the two open-loop approximations, how to obtain a new estimate of link travel time marginal distributions at each decision point? The computational tests derive the marginal distributions from the joint distribution, but this is not the case in reality.

3. Conduct more extensive computational tests to study the performance of the four approximations presented in this paper. The computational bottleneck

of the current computational tests is the generation of samples from the joint realization of all link travel times at all times. More efficient way is desired so that tests on larger network can be carried out.

4. Design approximations that are both realistic in traffic settings and computationally feasible. Test algorithms with real-world data. One possible approximation is the aggregate states approximation. It is also referred as feature extraction in dynamic programming. So far our network conditions have been in a very disaggregate level, i.e. each possible link travel time realization can possibly change the state. Sometimes, however, aggregate states could be used to reduce the dimension of the state space while still give a satisfactory representation of the network conditions. One possible aggregate state in traffic applications is the level of service, A, B, C, D, E, or F.

5. More comprehensive and rigorous analysis regarding the risk taking behavior is desired. The current discussion in Section 4.6 is based on the assumption of exponential disutility function. There are other forms of disutility functions that can be explored. Furthermore, there are other approaches other than disutility (utility) functions to take capture risk in the modeling of decision making under uncertainty.

Future research in the policy-based stochastic dynamic traffic assignment can be in the following directions:

1. In our current stochastic DTA model, the O-D trips are deterministic. However demand is one of the major sources of stochasticity in traffic systems. We need to extend our work to consider the demand stochasticity. Peeta and Zhou [23] proposed a way of considering demand stochasticity. We might develop our methods based on their work.

2. A more realistic dynamic network loading model is needed. Basically there are three types of dynamic network loading models. The first is the analytical model. This kind of models is able to provide mathematical properties of the

modeling system, but is generally not so realistic. The second is the simulation model. Realism is the outstanding characteristic of a simulation model, but it is usually very time-consuming. The third is the cell transmission model. This model is a combination of the previous two. It is relatively realistic, requires less computing resources and is suitable for implementing routing policies. We propose to use a cell transmission model in the future.

3. Efficient implementations of the policy-based stochastic DTA model are required to study the significance of stochasticity in a real network. High-performance computing implementations are considered.

4. DTA models with various assumptions on perception errors, risk taking behavior, information access, and network statistical dependency are to be developed. Our current model is a relatively simple one, as the main purpose is to show the concept. However, in order for the DTA model to be applicable, more realistic assumptions have to be made.

5. Traffic management is one of the primary application area of DTA models. Research on the design of traffic management schemes based on the developed DTA methods is essential for the application of DTA models. If implemented, the management scheme can provide a real world test bed for the DTA methods. This opportunity is invaluable.

6. Congestion has been our only target. However in an environmentally constrained world, traffic emission should be a problem as serious as congestion. DTA models considering both travel times and emissions are to be developed.

Appendix A

An Illustrative Example for Algorithm DOT-SPI

We use an example to illustrate how Algorithm DOT-SPI works. The small network in Figure A-1 has three nodes, three links and the number of time periods is 3. The values of the travel time realizations are in Table A.1. Each of the eight realizations has a probability of 0.125. The network is designed to be very small to make the understanding of the algorithm easier. Note that travelers starting from node 2 or node 3 have no choice but to take node 3 as the next node. It is suggested that the reader pay attention to how routing decision at node 1 is affected by time and information.

Step 0: Construct $\mathbf{EV}(t), t = 0, ..., 2$

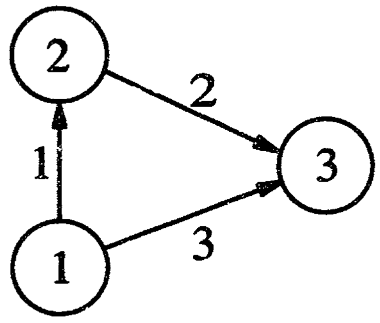


Figure A-1: Algorithm DOT-SPI: A Small Network

Time	Link	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
0	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1
	3	1	1	1	4	4	4	3	3
1	1	1	1	1	1	1	1	1	1
	2	2	2	1	2	2	1	2	1
	3	3	3	2	2	2	1	3	2
$t \geq 2$	1	1	1	2	1	1	1	2	2
	2	1	2	1	1	1	1	2	1
	3	3	2	2	3	4	3	5	2

Table A.1: Joint Realizations for the Small Network

Call Generate_Event_Collection

```
$D = \{\{v_1, ..., v_8\}\}$
$t = 0$
    $(j,k) = 1$
        $S = \{v_1, ..., v_8\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, ..., v_8\}\}$
        $D = \{\{v_1, ..., v_8\}\}$
    $(j,k) = 2$
        $S = \{v_1, ..., v_8\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, ..., v_8\}\}$
        $D = \{\{v_1, ..., v_8\}\}$
    $(j,k) = 3$
        $S = \{v_1, ..., v_8\}$
            $w = 3$
            $S'_1 = \{v_1, v_2, v_3\}, S'_2 = \{v_4, v_5, v_6\}, S'_3 = \{v_7, v_8\}$
            $D' = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$
            $D = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$
        $\mathbf{EV}(0) = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$
    $t = 1$
        $(j, k) = 1$
            $S = \{v_1, v_2, v_3\}$
                $w = 1$
                $S'_1 = S$
                $D' = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$
            $S = \{v_4, v_5, v_6\}$
                $w = 1$
                $S'_1 = S$
                $D' = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$
            $S = \{v_7, v_8\}$
                $w = 2$
                $S'_1 = \{v_7\}, S'_2 = \{v_8\}$
                $D' = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7\}, \{v_8\}\}$
            $D = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7\}, \{v_8\}\}$
        $(j, k) = 2$
            $S = \{v_1, v_2, v_3\}$
                $w = 2$
                $S'_1 = \{v_1, v_2\}, S'_2 = \{v_3\}$
                $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5, v_6\}, \{v_7\}, \{v_8\}\}$
            $S = \{v_4, v_5, v_6\}$
                $w = 2$
                $S'_1 = \{v_4, v_5\}, S'_2 = \{v_6\}$
                $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
            $S = \{v_7\}$
                $w = 1$
                $S'_1 = S$
                $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_8\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $D = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
$(j, k) = 3$
    $S = \{v_1, v_2\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_3\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_4, v_5\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_6\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_7\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $S = \{v_8\}$
        $w = 1$
        $S'_1 = S$
        $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $D = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $\mathbf{EV}(1) = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
$t = 2$
    $(j, k) = 1$
        $S = \{v_1, v_2\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_3\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_4, v_5\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_6\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_7\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_8\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $D = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $(j, k) = 2$
```

```
        $S = \{v_1, v_2\}$
            $w = 2$
            $S'_1 = \{v_1\}, S2' = \{v_2\}$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_3\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_4, v_5\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_6\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_7\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $S = \{v_8\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
        $D = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
    $(j, k) = 3$
        $S = \{v_1\}$
            $w = 1$
            $S'_1 = S$
            $D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$
```

$S = \{v_2\}$

$w = 1$

$S'_1 = S$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$S = \{v_3\}$

$w = 1$

$S'_1 = S$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$S = \{v_4, v_5\}$

$w = 2$

$S'_1 = \{v_4\}, S2' = \{v_5\}$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$S = \{v_6\}$

$w = 1$

$S'_1 = S$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$S = \{v_7\}$

$w = 1$

$S'_1 = S$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$S = \{v_8\}$

$w = 1$

$S'_1 = S$

$D' = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$D = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$\mathbf{EV}(2) = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

A summary of the results of constructing event collections is as follows.

$\mathbf{EV}(0) = \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}, \{v_7, v_8\}\}$

$\mathbf{EV}(1) = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

$\mathbf{EV}(2) = \{\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}, \{v_7\}, \{v_8\}\}$

Step 1: (Initialization)

1.1 Compute $e(j, 2, EV), \forall j \in N, \forall EV \in \mathbf{EV}(2)$

This step involves solving deterministic static shortest path problems with each single joint realization $v_r, r = 1, .., 8$. Any classical shortest path algorithm can be used. In our small network, this can be done by observation. The results are listed in Table A.2. In each result cell, the minimum expected travel time is given and the corresponding next node is in the parenthesis. We use "$n3$" to denote Node 3 and this rule of notation applies to all other nodes.

	$\{v_1\}$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_5\}$	$\{v_6\}$	$\{v_7\}$	$\{v_8\}$
$t = 3$	0 (n3)	0 (n3)	0 (n3)	0 (n3)	0 (n3)	0 (n3)	0 (n3)	0 (n3)
$t = 2$	1 (n3)	2 (n3)	1 (n3)	1 (n3)	1 (n3)	1 (n3)	2 (n3)	1 (n3)
$t = 1$	2 (n2)	2 (n3)	2 (n3)	2 (n2)	2 (n2)	2 (n2)	4 (n2)	2 (n3)

Table A.2: Results in the Static Deterministic Period

1.2 $e(j, t, EV) \leftarrow +\infty, \forall j \in N \backslash \{d\}$,

$e(d, t, EV) \leftarrow 0$,

$\forall t < K - 1, \forall EV \in \mathbf{EV}(t)$

Step 2: (Main Loop)

$t = 1$

$EV = \{v_1, v_2\}$

$EV_1' = \{v_1\}, Pr(EV_1'|EV) = 0.5$

$EV_2' = \{v_2\}, Pr(EV_2'|EV) = 0.5$

$(j, k) = (1, 2)$

$$temp = 1 + e(2, 1 + 1, EV_1')Pr(EV_1'|EV)$$
$$+e(2, 1 + 1, EV_2')Pr(EV_2'|EV)$$

$$= 1 + 1 \times 0.5 + 2 \times 0.5 = 2.5 < +\infty$$

$$e(1, 1, \{v_1, v_2\}) = 2.5, \mu^*(1, 1, \{v_1, v_2\}) = n2$$

$$(j, k) = (2, 3)$$

$$temp = 2 + e(3, 1 + 2, EV_1')Pr(EV_1'|EV)$$

$$+e(3, 1 + 2, EV_2')Pr(EV_2'|EV)$$

$$= 2 + 0 \times 0.5 + 0 \times 0.5 = 2 < +\infty$$

$$e(2, 1, \{v_1, v_2\}) = 2, \mu^*(2, 1, \{v_1, v_2\}) = n3$$

$$(j, k) = (1, 3)$$

$$temp = 3 + e(3, 1 + 3, EV_1')Pr(EV_1'|EV)$$

$$+e(3, 1 + 3, EV_2')Pr(EV_2'|EV)$$

$$= 3 + 0 \times 0.5 + 0 \times 0.5 = 3 > 2.5 = e(1, 1, \{v_1, v_2\})$$

$$EV = \{v_3\}$$

$$EV_1' = \{v_3\}, Pr(EV_1'|EV) = 1$$

$$(j, k) = (1, 2)$$

$$temp = 1 + e(2, 1 + 1, EV_1')Pr(EV_1'|EV)$$

$$= 1 + 1 \times 1 = 2 < +\infty$$

$$e(1, 1, \{v_3\}) = 2, \mu^*(1, 1, \{v_3\}) = n2$$

$$(j, k) = (2, 3)$$

$$temp = 1 + e(3, 1 + 1, EV_1')Pr(EV_1'|EV)$$

$$= 1 + 0 \times 1 = 1 < +\infty$$

$$e(2, 1, \{v_3\}) = 1, \mu^*(2, 1, \{v_3\}) = n3$$

$$(j, k) = (1, 3)$$

$$temp = 2 + e(3, 1 + 2, EV_1')Pr(EV_1'|EV)$$

$$= 2 + 0 \times 1 = 2 = e(1, 1, \{v_3\})$$

$$EV = \{v_4, v_5\}$$

$$EV_1' = \{v_4\}, Pr(EV_1'|EV) = 0.5$$

$$EV_2' = \{v_5\}, Pr(EV_2'|EV) = 0.5$$

$(j,k) = (1,2)$

$$temp = 1 + e(2, 1+1, EV_1')Pr(EV_1'|EV)$$
$$+e(2, 1+1, EV_2')Pr(EV_2'|EV)$$
$$= 1 + 1 \times 0.5 + 1 \times 0.5 = 2 < +\infty$$

$e(1,1,\{v_4,v_5\}) = 2, \mu^*(1,1,\{v_4,v_5\}) = n2$

$(j,k) = (2,3)$

$$temp = 2 + e(3, 1+2, EV_1')Pr(EV_1'|EV)$$
$$+e(3, 1+2, EV_2')Pr(EV_2'|EV)$$
$$= 2 + 0 \times 0.5 + 0 \times 0.5 = 2 < +\infty$$

$e(2,1,\{v_4,v_5\}) = 2, \mu^*(2,1,\{v_4,v_5\}) = n3$

$(j,k) = (1,3)$

$$temp = 2 + e(3, 1+2, EV_1')Pr(EV_1'|EV)$$
$$+e(3, 1+2, EV_2')Pr(EV_2'|EV)$$
$$= 2 + 0 \times 0.5 + 0 \times 0.5 = 2 = e(1,1,\{v_4,v_5\})$$

$EV = \{v_6\}$

$EV_1' = \{v_6\}, Pr(EV_1'|EV) = 1$

$(j,k) = (1,2)$

$$temp = 1 + e(2, 1+1, EV_1')Pr(EV_1'|EV)$$
$$= 1 + 1 \times 1 = 2 < +\infty$$

$e(1,1,\{v_6\}) = 2, \mu^*(1,1,\{v_6\}) = n2$

$(j,k) = (2,3)$

$$temp = 1 + e(3, 1+1, EV_1')Pr(EV_1'|EV)$$
$$= 1 + 0 \times 1 = 1 < +\infty$$

$e(2,1,\{v_6\}) = 1, \mu^*(2,1,\{v_6\}) = n3$

$(j,k) = (1,3)$

$$temp = 1 + e(3, 1+1, EV_1')Pr(EV_1'|EV)$$
$$= 1 + 0 \times 1 = 1 < 2 = e(1,1,\{v_6\})$$

$e(1,1,\{v_6\}) = 1, \mu^*(1,1,\{v_6\}) = n3$

$EV = \{v_7\}$

$EV'_1 = \{v_7\}, Pr(EV'_1|EV) = 1$

$(j,k) = (1,2)$

$temp = 1 + e(2, 1+1, EV'_1)Pr(EV'_1|EV)$

$= 1 + 2 \times 1 = 3 < +\infty$

$e(1,1,\{v_7\}) = 3, \mu^*(1,1,\{v_7\}) = n2$

$(j,k) = (2,3)$

$temp = 2 + e(3, 1+2, EV'_1)Pr(EV'_1|EV)$

$= 2 + 0 \times 1 = 2 < +\infty$

$e(2,1,\{v_7\}) = 2, \mu^*(2,1,\{v_7\}) = n3$

$(j,k) = (1,3)$

$temp = 3 + e(3, 1+3, EV'_1)Pr(EV'_1|EV)$

$= 3 + 0 \times 1 = 3 = e(1,1,\{v_7\})$

$EV = \{v_8\}$

$EV'_1 = \{v_8\}, Pr(EV'_1|EV) = 1$

$(j,k) = (1,2)$

$temp = 1 + e(2, 1+1, EV'_1)Pr(EV'_1|EV)$

$= 1 + 1 \times 1 = 2 < +\infty$

$e(1,1,\{v_8\}) = 2, \mu^*(1,1,\{v_8\}) = n2$

$(j,k) = (2,3)$

$temp = 1 + e(3, 1+1, EV'_1)Pr(EV'_1|EV)$

$= 1 + 0 \times 1 = 1 < +\infty$

$e(2,1,\{v_8\}) = 1, \mu^*(2,1,\{v_8\}) = n3$

$(j,k) = (1,3)$

$EV'_1 = \{v_8\}, Pr(EV'_1|EV) = 1$

$$\begin{aligned} temp &= 2 + e(3, 1+2, EV_1')Pr(EV_1'|EV) \\ &= 2 + 0 \times 1 = 2 = e(1, 1, \{v_8\}) \end{aligned}$$

The summary of results at time 1 is in Table A.3.

	$\{v_1, v_2\}$	$\{v_3\}$	$\{v_4, v_5\}$	$\{v_6\}$	$\{v_7\}$	$\{v_8\}$
$t = 3$	0(n3)	0(n3)	0(n3)	0(n3)	0(n3)	0(n3)
$t = 2$	2(n3)	1(n3)	2(n3)	1(n3)	2(n3)	1(n3)
$t = 1$	2.5(n2)	2(n2)	2(n2)	1(n3)	3(n2)	2(n2)

Table A.3: Results at Time 1

$t = 0$

$EV = \{v_1, v_2, v_3\}$

$EV_1' = \{v_1, v_2\}, Pr(EV_1'|EV) = 2/3$

$EV_2' = \{v_3\}, Pr(EV_2'|EV) = 1/3$

$(j, k) = (1, 2)$

$$\begin{aligned} temp &= 1 + e(2, 0+1, EV_1')Pr(EV_1'|EV) \\ &+e(2, 0+1, EV_2')Pr(EV_2'|EV) \\ &= 1 + 2 \times 2/3 + 1 \times 1/3 = 8/3 < +\infty \end{aligned}$$

$e(1, 0, \{v_1, v_2, v_3\}) = 8/3, \mu^*(1, 0, \{v_1, v_2, v_3\}) := n2$

$(j, k) = (2, 3)$

$$\begin{aligned} temp &= 1 + e(3, 0+1, EV_1')Pr(EV_1'|EV) \\ &+e(3, 0+1, EV_2')Pr(EV_2'|EV) \\ &= 1 + 0 \times 2/3 + 0 \times 1/3 = 1 < +\infty \end{aligned}$$

$e(2, 0, \{v_1, v_2, v_3\}) = 1, \mu^*(2, 0, \{v_1, v_2, v_3\}) = n3$

$(j, k) = (1, 3)$

$$\begin{aligned} temp &= 1 + e(3, 0+1, EV_1')Pr(EV_1'|EV) \\ &+e(3, 0+1, EV_2')Pr(EV_2'|EV) \\ &= 1 + 0 \times 2/3 + 0 \times 1/3 = 1 < 8/3 = e(1, 0, \{v_1, v_2, v_3\}) \end{aligned}$$

$$e(1,0,\{v_1,v_2,v_3\}) = 1, \mu^*(1,0,\{v_1,v_2,v_3\}) = n3$$

$EV = \{v_4, v_5, v_6\}$

$EV_1' = \{v_4, v_5\}, Pr(EV_1'|EV) = 2/3$

$EV_2' = \{v_6\}, Pr(EV_2'|EV) = 1/3$

$(j,k) = (1,2)$

$$temp = 1 + e(2, 0+1, EV_1')Pr(EV_1'|EV)$$
$$+e(2,0+1,EV_2')Pr(EV_2'|EV)$$
$$= 1 + 2 \times 2/3 + 1 \times 1/3 = 8/3 < +\infty$$
$$e(1,0,\{v_4,v_5,v_6\}) = 8/3, \mu^*(1,0,\{v_4,v_5,v_6\}) = n2$$

$(j,k) = (2,3)$

$$temp = 1 + e(3, 0+1, EV_1')Pr(EV_1'|EV)$$
$$+e(3,0+1,EV_2')Pr(EV_2'|EV)$$
$$= 1 + 0 \times 2/3 + 0 \times 1/3 = 1 < +\infty$$
$$e(2,0,\{v_4,v_5,v_6\}) = 1, \mu^*(2,0,\{v_4,v_5,v_6\}) = n3$$

$(j,k) = (1,3)$

$$temp = 4 + e(3, 0+4, EV_1')Pr(EV_1'|EV)$$
$$+e(3,0+4,EV_2')Pr(EV_2'|EV)$$
$$= 4 + 0 \times 2/3 + 0 \times 1/3 = 4 > 8/3 = e(1,0,\{v_4,v_5,v_6\})$$

$EV = \{v_7, v_8\}$

$EV_1' = \{v_7\}, Pr(EV_1'|EV) = 0.5$

$EV_2' = \{v_8\}, Pr(EV_2'|EV) = 0.5$

$(j,k) = (1,2)$

$$temp = 1 + e(2, 0+1, EV_1')Pr(EV_1'|EV)$$
$$+e(2,0+1,EV_2')Pr(EV_2'|EV)$$
$$= 1 + 2 \times 0.5 + 1 \times 0.5 = 2.5 < +\infty$$
$$e(1,0,\{v_7,v_8\}) = 2.5, \mu^*(1,0,\{v_7,v_8\}) = n2$$

$(j, k) = (2, 3)$

$$\begin{aligned} temp &= 1 + e(3, 0+1, EV_1')Pr(EV_1'|EV) \\ &\quad + e(2, 0+1, EV_2')Pr(EV_2'|EV) \\ &\quad = 1 + 0 \times 0.5 + 0 \times 0.5 = 1 < +\infty \end{aligned}$$

$e(2, 0, \{v_7, v_8\}) = 1, \mu^*(2, 0, \{v_7, v_8\}) = n3$

$(j, k) = (1, 3)$

$$\begin{aligned} temp &= 3 + e(3, 0+3, EV_1')Pr(EV_1'|EV) \\ &\quad + e(3, 0+3, EV_2')Pr(EV_2'|EV) \\ &\quad = 3 + 0 \times 0.5 + 0 \times 0.5 = 3 > 2.5 = e(1, 0, \{v_7, v_8\}) \end{aligned}$$

The summary of results at time 0 is in Table A.4.

	$\{v_1, v_2, v_3\}$	$\{v_4, v_5, v_6\}$	$\{v_7, v_8\}$
$t = 3$	0(n3)	0(n3)	0(n3)
$t = 2$	1(n3)	1(n3)	1(n3)
$t = 1$	1(n3)	8/3(n2)	2.5(n2)

Table A.4: Results at Time 2

Bibliography

[1] R. Ahuja, T. Magnanti, and Orlin J. *Network Flows.* Prentice-Hall, New Jersey, 1993.

[2] G. Andreatta and L. Romeo. Stochastic shortest paths with recourse. *Networks*, 18:193–204, 1988.

[3] M. E. Ben-Akiva and S. R. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand.* MIT Press, Cambridge, MA, 1985.

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, Massachusetts, 2nd edition, 2000.

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, Massachusetts, 1996.

[6] J. A. Bottom. *Anticipatory Consistent Route Guidance.* PhD thesis, Massachusetts Institute of Technology, September 2000.

[7] G. E. Cantarella and E. Cascetta. Dynamic processes and equilibrium in transportation networks: Toward a unifying theory. *Transportation Science*, 29(4):305–329, 1995.

[8] E. Cascetta and G. E. Cantarela. A day-to-day and within-day dynamic stochastic assignment model. *Transportation Research Part A*, 25(5):277–291, 1991.

[9] I. Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record*, 1645:170–175, 1999.

[10] I. Chabini. Minimum expected travel times in stochastic time-dependent networks revisited. Internal Report. MIT, Cambridge, MA, 2000.

[11] I. Chabini and Y. He. An analytical approach to dynamic traffic assignment: Models, algorithms, and computer implementations. Accepted for Publication in *International Transactions on Operations Research*, 2000.

[12] I. Chabini and S. Lan. Considering the spill-back of queues in dynamic network loading model. Internal Report. MIT, Cambridge, MA, 2000.

[13] R. K. Cheung. Iterative methods for dynamic stochastic shortest path problems. *Navel Research Logistics*, 45:769–789, 1998.

[14] J. Croucher. A note on the stochastic shortest-route problem. *Naval Research Logistics Quarterly*, 25:729–732, 1978.

[15] C. F. Daganzo and Y. Sheffi. On stochastic models of traffic assignment. *Transportation Science*, 11:253–274, 1977.

[16] L. Fu and L. R. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. *Transportation Research Part B*, 32(7):35–52, 1998.

[17] R. W. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):91–101, 1986.

[18] M. Hazelton. Some remarks on stochastic user equilibrium. *Transportation Research Part B*, 32(2):101–108, 1998.

[19] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, 1976.

[20] E. D. Miller-Hooks and H. S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, 2000.

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

[22] P. Mirchandani and H. Soroush. Generalized traffic equilibrium with probabilistic travel times and perceptions. *Transportation Science*, 21(3), 1987.

[23] S. Peeta and C. Zhou. Robustness of the off-line a priori stochastic dynamic traffic assignment solution for on-line operations. *Transportation Research Part C*, 7:281–303, 1999.

[24] G. Polychronopoulos. *Stochastic and Dynamic Shortest Distance Problems.* PhD thesis, Massachusetts Institute of Technology, May 1992.

[25] G. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27:133–143, 1996.

[26] H. N. Psaraftis and J. N. Tsitsiklis. Dynamic shortest paths in acyclic networks with markovian arc costs. *Operations Research*, 41(1):91–101, 1993.

[27] Y. Sheffi. *Urban Transportation Networks.* Prentice-Hall, Englewood, NJ, 1985.