The Thesis Committee for Tarun Rambha

Certifies that this is the approved version of the following thesis:

Adaptive Routing in Schedule Based Stochastic Time-Dependent Transit Networks

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor: _____

Stephen D. Boyles

Co-supervisor: _____

S. Travis Waller

Adaptive Routing in Schedule Based Stochastic Time-Dependent Transit Networks

by

Tarun Rambha, B.Tech

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2012

To my parents

Acknowledgments

I take this opportunity to express my gratitude to my advisors Dr.Stephen Boyles and Dr.Travis Waller. They have been truly inspiring in their own right. In particular, I would like to thank Dr.Stephen Boyles for countless discussions on the various issues that came up during the course of this work and for pointing out the analogy between light cones and the proposed methodologies. I thank Dr.Waller for his encouragement during the initial stages of my graduate studies and for giving me the time and freedom to explore a topic of interest for my thesis.

I wish to thank Natalia Ruiz, Nezamuddin, Jennifer Duthie, Christopher Melson and Shoupeng Tang with whom I enjoyed working with as a graduate research assistant. In addition, I thank Lisa Cramer and Libbie Toler for their timely administrative assistance. I appreciate the efforts of faculty at UT particularly, Dr.Anath Balakrishnan, Dr.David Morton, Dr.Thomas Wiseman and Dr.Chandra Bhat whose courses were intellectually stimulating and helped reinforce my interests. I would also like to thank Dr.Karthik Srinivasan for introducing me to this wonderful field back at IIT Madras.

My graduate experiences were enriched by the company of several friends whom I wish to acknowledge; Roshan and Rajesh for being great mentors; GT for discussions on various aspects of life; Prasad and Sundeep for comic relief; and Raghu, Pallu, Ravi, Sriram, Moby, Gaurav, Chrissy and Jaggu for making my stay at Austin enjoyable. Also, I would like to thank all forces of nature that left me with no choice but to walk for half an hour and wait for the next show at a movie theater after missing a transfer between buses on routes 10 and 311 on one fateful day. This incident later motivated the thought of using more information to define better adaptive strategies.

I am greatly indebted to my parents, grandmother and sister for their love and sacrifices and for encouraging my pursuits. Lastly, I thank god for his grace and blessings.

Adaptive Routing in Schedule Based Stochastic Time-Dependent Transit Networks

Tarun Rambha, MSE
The University of Texas at Austin, 2012

Supervisor: Stephen D. Boyles
Co-supervisor: S. Travis Waller

In this thesis, an adaptive transit routing (ATR) problem in a schedule based stochastic time-dependent transit network is defined and formulated as a finite horizon Markov Decision Process (MDP). The transit link travel times are assumed to be random with known probability distributions. Routing strategies are defined to be conditional on the arrival times at intermediate nodes, and the location and arrival times of other buses in the network. In other words, a traveler in the network decides to walk, wait or board a bus based on the real time information of all buses in the network. The objective is to find a strategy that minimizes the expected travel time, subject to constraints that guarantee that the destination is reached within a certain threshold. The value of the threshold was chosen to reflect the risk averse attitude of travelers and is computed based on the earliest time by which the destination can be reached with probability 1. The problem inherits *the curse of dimensionality* and state space reduction through pre-processing is achieved by solving variants of the time dependent shortest path problem. An interesting analogy between the state space reduction techniques and the concept of light cones is discussed. A dynamic program framework to solve the problem is developed by defining the state space, decision space and transition functions. Numerical results on a small instance of the Austin transit network are presented to investigate the extent of reduction in state space using the proposed methods.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Transportation networks are often subject to uncertainty in link travel times. Variability in the time taken to traverse a link could result from several factors such as congested road conditions, presence of traffic signals, inclement weather, etc. Since transit networks (particularly bus networks) use roadway links, the time taken for travel in these networks is also pervaded by uncertainty. While randomness in transit networks might not have a bite in a traveler's route choice when he/she has no option but to board a single bus, it does play an important role for trips that involve transfers and in cities with a large number of alternate transit options for traveling between an origin-destination (OD) pair. However, most transit routing applications seldom take this into account while providing routing policies. The a priori strategies prescribed by these applications inform travelers where to board, get down or transfer based on frequencies or schedules published by transit agencies.

1.2 Motivation

A priori strategies in stochastic networks are generally sub-optimal and by making use of additional information, it is possible to construct adaptive strategies that are optimal. Advances in Intelligent Transportation Systems (ITS) let us gather a large amount of real time data related to location and travel time of buses in a network which may be used for this purpose. This information also plays a vital role in characterizing the distributions of uncertainty in the network. For instance, suppose a traveler's a priori strategy is to board bus A and transfer to another bus B at

some node in the network. If either of the two buses gets delayed, the possibility of a transfer and the waiting time (if a transfer is feasible) can affect the optimality of the a priori strategy. In such cases the travelers routing policy can be modified dynamically using information related to the real time location of buses in the network. Thus, the problem of traveling in transit networks in optimal time can be regarded as being adaptive in nature, where decisions made at nodes are conditional on the location of buses.

Consider a small example that illustrates the adaptive nature of the problem. Suppose, buses A and B start at nodes 1 and 4 at $t = 0$ respectively, in the network shown in figure 1.1. The time taken by the buses to traverse the transit arcs is indicated in the figure. The travel time on the transit arc (4,5) may be 1 or 10 with equal probability. Also, let the walking travel time on arcs shown in the network be 5.

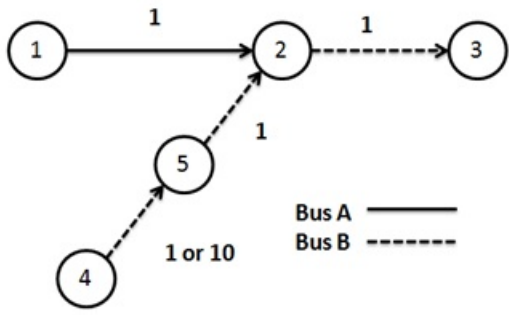

Figure 1.1: Illustration of adaptive routing in transit networks

Suppose we wish to travel from node 1 to node 3. If we board bus A and reach node 2 at time $t = 1$, we can either walk or wait for bus 2. Hence, we have two a priori strategies whose expected costs are shown in the table 1.1. If we decide to walk to node 3 after boarding bus A, we would incur a cost of 1(time taken by bus A to reach node 2) + 5 (walking travel time between nodes 2 and 3) = 6. On the other hand, if we wait for bus B, the destination is possibly reached at $t = 3$ or $t = 12$ with equal

probability. Hence, the expected cost of travel is 3(0.5) + 12(0.5) = 7.5. Therefore, the optimal a priori strategy is to board bus A and walk to the destination.

Table 1.1: A priori strategies

Strategy	*Expected cost*
Board bus A and walk to node 3	6
Board bus A and wait for bus B	7.5

But if bus B reaches node 5 at $t = 1$, it is guaranteed to arrive at node 2 at $t = 2$ and hence waiting is optimal. However, if we received information that bus B failed to reach node 5 at $t = 1$, walking from node 2 to node 3 is optimal. Such an adaptive strategy would have an expected cost of 3(0.5) + 6(0.5) = 4.5, which is lower than the optimal a priori solution.

1.3 Objectives

This research is applicable to schedule based transit networks which are subject to uncertainty in link travel times with known pmfs. A major goal of this thesis is to develop an adaptive route choice model in the presence of real time information. In defining an adaptive strategy we use the notion of system states, which are characterized by the spatial and temporal locations of buses and the traveler in the network. Travelers are assumed to choose least expected cost strategies which guarantee that the destination is reached within a given threshold. Using this backdrop, the central idea in finding an optimal strategy can be described as follows. For each choice available to a traveler at a particular state, the expected time to reach the destination from possible future states can be used to determine the optimal decision at that state. Given below is a list of the major objectives of this study.

1. *Defining the Adaptive Transit Routing(ATR) problem:*
   This step mainly involves network transformations and creation of the state space of buses in the network using the travel time distributions on links.

2. *Reducing the size of the state space:*

   Since buses in the network can possibly be at different stops in the network at various times, the size of the resulting state space is usually very large. For instance, if a bus is scheduled to traverse 20 links on a particular trip and if the size of the support of the travel time distributions on each link is 2, the number of states at the last node is of the order $2^{20} \approx 10^6$. This calls for algorithms and preprocessing methods to refine the state space in order to ease the computation of the optimal strategy. In addition, numerical experiments to determine the effectiveness of these methods are to be conducted.

3. *Development of a dynamic programming framework:*

   This involves formulating the problem of finding the optimal strategy as a Markov Decision Process (MDP) using elements such as state space, decision space, transition and value functions.

1.4 Organization of thesis

The rest of this thesis is organized as follows. The second chapter reviews existing literature on adaptive shortest paths under uncertainty and route choice in transit networks. Chapter 3 comprises of a description of the problem and the notation used. Chapter 4 develops algorithms used for preprocessing and examines the elimination of individual state space in detail. Chapter 5 explains the framework for solving the ATR problem as an MDP and suggests possible solution methods. Chapter 6 contains the computational results of an implementation of the state space reduction method on a small instance of the Austin transit network and summarizes the findings and limitations of this study, and discusses possible directions for future research.

Chapter 2

Literature Review

2.1 Introduction

Shortest path problems in public transportation systems have been of interest primarily for a couple of reasons. First, it can assist a traveler in the decision making process by optimizing his/her objectives. Secondly, knowing how people choose routes can be used to obtain transit flows by solving a transit assignment problem. This may further be applied in transit planning as it forms the basis for network design and scheduling problems.

At the outset, finding shortest paths in transit networks may appear to be similar to optimal routing in general traffic networks. However, the problem is relatively difficult due to the possibility of waiting and the issue of *common bus lines* as noted by Chriqui and Robillard [4]. A traveler at a node in a transit network is often faced with the option of boarding multiple buses on different lines to traverse a link or a section of a route. Also, presence of randomness in travel times and arrivals of buses in the network adds an extra layer of complexity to the problem. These features have motivated researchers in the past to develop adaptive routing algorithms for transit networks.

Based on its applications, literature on adaptive shortest paths in transit networks can be classified into stand-alone adaptive routing problems and adaptive transit route choice in traffic assignment. While the former inspired research related to stochastic and online shortest paths in general traffic networks, studies on the latter expanded to

include passenger capacity constraints, effects of queues, etc. Although, the objectives of a traveler in these classes of problems are similar in most cases, subtle differences in the underlying assumptions distinguish them from each other. For example, more emphasis is laid on probabilistic link travel times in regular adaptive routing problems. On the other hand, adaptive route choice models in traffic assignment seek an accurate representation of waiting times under uncertainty in arrivals/headways of buses.

In the following sections, we review adaptive and online shortest paths in stochastic time varying networks, transit route choice in the context of transit assignment, trip planning models in the presence of real time information and a few other approaches for transit routing. Finally, we position the contribution of this thesis with respect to existing literature.

2.2 Adaptive routing in stochastic networks

The standard shortest path problem with deterministic non negative arc costs can be solved by using the well-known label setting algorithm proposed by Dijkstra [7]. The Bellman-Ford algorithm (which falls under the category of a label correcting algorithm), originally suggested by Ford [9], and independently by Bellman [2] can handle a more general class of shortest path problems. Over the last few decades, these problems have been extensively studied and several efficient implementations were developed, details of some of which can be found in Ahuja et al. [1], and Deo and Pang [6].

However, in most transportation networks, the travel time on an arc is not fixed, but is dictated by the time at which one arrives at its tail node. Such a property is evident in transit networks as the waiting time is a function of the arrival time of buses. Dreyfus [8] extended Dijkstra's algorithm to solve for the shortest path in networks with time dependent arc costs in which certain FIFO assumptions hold. Ziliaskopolous and

Mahmassani [32] provide a label correcting approach for the time dependent shortest path (TDSP) problem by maintaining multiple labels at each node. A comprehensive summary of TDSP algorithms and their computational complexities can be found in Chabini [3].

In networks with probabilistic and independent arc costs, the least expected time path can be computed by formulating it as a static shortest path problem in which the arc costs are replaced by their expected values. However, if the link travel times are random and time dependent, or if the pmfs vary with time, standard shortest path algorithms fail to find the optimal path. In fact, the optimal choice in such networks is not a path but a strategy/hyperpath or a complete contingent plan of action. This was first pointed out by Hall [11] in his seminal paper on this subject. It was shown that a path chosen based on the arrival times at intermediate nodes might result in lesser expected travel time. An algorithm for an a priori strategy and a dynamic programming approach for finding the adaptive policy were presented and demonstrated using a transit network.

Miller-Hooks and Mahmassani [19] developed modified label correcting algorithms for finding the a priori least expected cost path and lower bounds on the expected travel time of the optimal adaptive strategy in a network in which the pmfs of the arc costs are independent and time varying. The latter was extended by Miller-Hooks [18], and efficient modified label setting and correcting methods for finding the optimal adaptive policy were suggested. Similar to the approach followed by Hall [11], strategies were assumed to be conditional on the arrival time at intermediate nodes. The performance of these algorithms was tested on transportation and data networks. Based on shortest path algorithms in hypergraphs, derived by Nguyen and Pallottino [21], Pretolani [24] solved a similar problem of finding the shortest adaptive path using weighted hypergraphs.

Other variants of shortest path problems in stochastic networks are derived by modifying the assumptions of the source of uncertainty and the timing of observation. Polychronopoulos and Tsitsiklis [22] formulated stochastic shortest path problems with recourse using a dynamic programming framework in which, arc costs are random with known distributions and the uncertainty is partly revealed as the network is traversed (i.e. when a traveler reaches the tail node of an arc). The authors handle both spatial dependence and independence among arc costs by constructing possible realizations of the network. Among other related works, Waller and Ziliaskopoulos [29], and Provan [25] extended the problem of finding adaptive strategies in networks with arc cost dependencies under a reset assumption according to which, the arc cost is realized every time its tail node is reached even if it was previously visited.

2.3 Adaptive routing in transit assignment

Finding the optimal adaptive route appears as a sub-problem in the transit assignment problem. These problems have been studied in great detail and are usually centered around frequency based transit networks. Most authors also presume that the headway between bus arrivals is random with known distributions (typically exponential). In the presence of common bus lines, travelers are assumed to choose a subset of available lines, also called as the *attractive set*, such that the total expected travel time to the destination in minimized. A strategy is defined by the line chosen at each stop (or a probability distribution over the attractive set) and the alighting point given that a particular line was chosen. The optimal routing policy is then employed to obtain transit flows by solving for equilibrium using either time dependent OD demands or by assuming probabilistic passenger arrivals.

Spiess and Florian [26], and Nguyen and Pallottino [20] were among the first to make notable contributions to the assignment problem. The former consider a transit network in which transit travel times are fixed but the waiting time at nodes depend

on the combined frequency of lines. They assume that the strategy of a traveler is to board the first bus serving a line that belongs to the attractive set. A linear programming formulation was used to find the optimal strategy, which was further extended to model instances in which the in-vehicle travel time varies as a function of passenger flows (referred to as *discomfort functions*) using a non-linear program. The authors briefly discuss other possible strategies depending on the information available during the course of travel. One such suggestion alludes to the idea of defining strategies based on information acquired on buses spotted while riding a bus. A manifestation of this thought is in a way similar to the concept of a strategy used in this thesis, in which actions are conditional on the spatial and temporal information of buses in the network.

Nguyen and Pallottino [20] on the other hand, used a graph theoretical framework, in which travelers are assumed to choose hyperpaths instead of transit arcs. The hyperpath of a traveler defines a strategy and provides the probability of boarding a line that belongs to the attractive set. The cost of a hyperapth includes fixed transit arc costs and waiting travel times weighted by appropriate probabilities. The authors then use a variational inequality to formulate the transit assignment problem based on the shortest hyperpaths chosen by travelers.

Broader classes of transit assignment problems which incorporate the effects of congestion have also been examined by several researchers. Cea and Fernandez [5] analyzed the transit assignment problem in which a traveler boards the first bus from the set of attractive lines only if the capacity of the bus is not exceeded. Wu et al. [30] used the concept of hyperpaths and extended the non-linear model of Spiess and Florian [26] to include waiting times that depend on flows and frequencies in addition to the discomfort functions. Gentile et al. [10] find the optimal strategy and the equilibrium transit flows in the presence of online information at stops. Passengers are provided

with estimates of waiting time for each line and are assumed to have knowledge of the total expected time to reach the destination (which may include waiting and in-vehicle travel time for subsequent transfers) after choosing a particular line.

Strategies in transit assignment, in public transportation systems based on schedules have received limited attention in literature. The waiting time in these problems is a function of the arrival time of buses. Tong and Richardson [28] formulated the problem of finding the optimal path in a transit network with schedules as a TDSP problem using weighted costs of trip components. A branch and bound type algorithm was developed to search for the shortest route.

Hamdouch and Lawphongpanich [12] present a transit assignment model with capacity constraints using travel strategies in a time expanded network. At each node except the destination, a subset of downstream nodes (called the user-preference set) ordered by their preference of being chosen is considered. A collection of user-preference sets at all nodes in the network induces an acyclic subgraph or a strategy subgraph. Given a particular strategy, the probability with which an arc is traversed or a node is visited is calculated using the number of passengers assigned to all possible strategies. The expected cost of a strategy is computed and a variational inequality formulation is used to find the flows and the optimal strategies for each OD pair.

2.4 Transit trip planning using real time data

Significant contributions to route choice in stochastic time-dependent transit networks in the presence of online information were made by Hickman [13], Hickman and Wilson [16], and Hickman and Bernstein [15]. Two path choice models (static and dynamic) were proposed for a schedule based public transportation network. Travel times of trips are assumed to be governed by known continuous probability distributions. In the static path choice model, a traveler chooses a set of paths, similar to the attractive

set in the common bus line problem. On the other hand, the dynamic path choice model analyzes if a traveler at the origin has to board a bus or wait for a bus that arrives later based on the information (i.e. actual departure times of buses at their source nodes and the arrival time of the traveler at his/her origin) gained while waiting. Simulation experiments on a small corridor were shown to exhibit modest improvements in travel time but significant changes in optimal paths.

Hickman [14] suggested the use of historical real time data to obtain probability distributions of transit arcs and consequently, the cumulative distributions of travel time on a path. In addition, a procedure for generating paths and pruning dominated ones based on the notion of stochastic dominance was studied.

2.5 Other transit routing approaches

In finding an optimal scheme, a variety of objectives different from minimizing cost may be considered. For instance, a traveler might be interested in a trip plan with fewer transfers or less walking/waiting. In fact, an increasing number of web based services (such as Google Maps, websites of local transit agencies etc.) have begun to incorporate such objectives in their route planning applications.

These problems have been widely addressed in the literature for networks with fixed schedules. Tan et al. [27] describe a procedure to find the shortest route which satisfies constraints on the arrival time and the walking distances. Huang and Peng [17] developed forward and backward search algorithms for finding the optimal path for a given departure time and an expected arrival time respectively. Xu et al. [31] devised efficient algorithms for finding the K shortest paths in a schedule based transit system. However, none of these approaches consider the effects of travel time uncertainty and hence the routing plans are not adaptive in nature.

2.6 Summary

In this chapter, we surveyed literature on adaptive routing and its applications in transit networks. Three major classes of problems, relevant to the current thesis, were discussed at length. These include adaptive routing in regular traffic networks, optimal paths in the context of transit assignment and trip planning in the presence of real time information.

As mentioned earlier, strategies in transit assignment models are defined with regard to frequency based public transportation systems unlike the schedule based approach used in this thesis. These models focus on the estimating the expected waiting time and assume deterministic vehicle travel times. Also, transfers are not explicitly modeled. At this juncture, we reserve our comparisons of these studies with the current work recognizing the fact that routing is a sub problem to the assignment problem, which limits the extent of complexity that can be introduced in defining strategies.

Adaptive routing methods discussed in section 2.2 do extend the concept of a strategy, but applications in literature on this topic (except for the ones presented by Hall [11]) are tailored to general traffic networks. Also, making decisions conditioned on the arrival time at a particular node is not necessarily optimal in transit networks as seen in section 1.2.

Research by Hickman [13], Hickman and Wilson [16], and Hickman and Bernstein [15] investigate strategies in a stochastic time-dependent network with online information which is to an extent similar to the ATR problem. Conceptually, their models can be extended to handle situations involving transfers, in which strategies are not only dependent on the arrival time at a node (the origin or transfer point), but also on the information of buses serving the node, received until that node is reached. Yet, their adaptive framework is still myopic in nature as the proposed extension considers only

the information of buses that arrive at a particular node at which the traveler boards or transfers. Further, the use of continuous probability distributions raises concerns about a possible implementation or development of an analytical solution for large networks.

In this thesis, we attempt to develop an adaptive route choice model using a more unrestrictive definition of a strategy, in the presence of vast information on all buses in the network. A treatment of the ATR problem as an MDP is facilitated by the definitions of the state of the system and the discretization mechanism. However, the problem size can grow exponentially and hence more emphasis is laid on the reduction of the state space in order to improve the tractability of the dynamic program.

Chapter 3

Problem Description

In this chapter, we first represent the public transportation network as a graph with walking and transit arcs and present some useful notation. Assumptions used in the ATR problem are outlined and are compared to those used by other authors followed by a demonstration of the process of constructing the individual state space of buses using a few examples. Finally, the objectives of a traveler and measures of risk aversion are briefly discussed.

3.1 Notation

Let $G(N, A)$ be a directed network, where N represents the set of nodes/bus stops. A node which is the destination of a route and the origin of another is replicated (explained in detail using an example in section 3.3). The set of arcs A is defined as $A_w \cup A_r \cup A_d$, where A_w represents the set of shortest walking arcs between every pair of nodes in N, A_r consists of links between bus stops along routes in the network and A_d represents the set of dummy arcs used to connect node copies to model the slack in schedules. Let $b \in B$ and $r \in R$ represent the set of buses and routes respectively. The set of routes in the network is similar to those defined by transit agencies, except that routes in opposite directions (for instance northbound (NB) and southbound (SB)) are treated as different routes. The time period of interest T is divided into unit intervals $\{0, 1, 2, .., t, ..|T| - 1\}$ each of which denotes the time elapsed from a fixed time (say the start of the first trip of first bus). For example, if the first bus enters the network at say 6:00 AM, then 7:20 AM is represented as $t = 120$. Let t_O be the time at which a traveler departs at the origin node.

Table 3.1: List of symbols

Symbol	*Description*
$r \in R$	Set of Routes
$b \in B$	Set of Buses
$\rho_r \in P_r$	Set of trips along route r. The k^{th} trip on a route is denoted by ρ_r^k
$\mathbb{M}_r : P_r \to B$	A mapping which assigns a bus to each trip on route r
$\beta_b \in I_b$	Itinerary of a bus b which is the set of trips made by the bus. A trip is assumed to contain information related to the stops and the scheduled arrival times. We reference the k^{th} trip in I_b by β_b^k
$n_{\beta_b} \in N_{\beta_b}$	Set of nodes visited by bus b in trip β_b. We reference the k^{th} node in trip β_b by $n_{\beta_b}^k$
$t_{n_{\beta_b}} \in T_{n_{\beta_b}}$	Set of times at which a bus reaches node n_{β_b}
$f_{n_{\beta_b}}$	Earliest possible time a bus b can reach node n_{β_b}
$l_{n_{\beta_b}}$	Latest possible time a bus b can reach node n_{β_b}
$a_{\beta_b} \in A_{\beta_b}$	Set of arcs included in a trip β_b
$\Omega_{a_{\beta_b}}$	Distribution of time on arc a_{β_b} (Without loss of generality assume the pmf is arranged in increasing order of time)
$C^{\omega}_{a_{\beta_b}}$	A particular realization of the cost on arc a_{β_b}
w_{ij}	Walking arc cost between nodes i and j, where $i, j \in N$
$order(s_b)$	The number of nodes visited by bus b from the current state before reaching the state s_b
$\tilde{n}(s_b)$	Node associated with state s_b, where $b \in B$
$\tilde{t}(s_b)$	Time associated with state s_b, where $b \in B$

We define the set of *individual states* of a bus $s_b \in S_b$ using the ordered pair $(n_{\beta_b}, t_{n_{\beta_b}})$, where n_{β_b} denotes the most recently visited bus stop and $t_{n_{\beta_b}}$ is the time at which the bus b arrived at node n_{β_b}. The individual state of buses, in practice may be determined by observing the actual arrival times of buses at bus stops in the network. The *system state space* S is defined as the subset of the cartesian product of the individual states, the set of nodes N and time periods T, i.e. $S \subseteq \mathbb{S}$ where $\mathbb{S} = S_{b_1} \times S_{b_2} \times S_{b_3} ... \times S_{b_{|B|}} \times N \times T$. The construction of the individual states S_b and

the state space S will be discussed later. The node n and time t in a state vector $(s_{b_1}, s_{b_2}, ...s_{b_{|B|}}, n, t)$ denotes the node and time where a traveler is present in the network (also referred as *individual state of the traveler*).

3.2 Assumptions

Given below is a list of assumptions used in the ATR problem. These assumptions not only assist in defining the states of the buses in the network but are also pivotal to some of the assertions made in algorithms and preprocessing methods discussed in later chapters.

1. Each bus can serve multiple routes. However, a bus can serve route r_1 followed by route r_2, only if the final destination of r_1 is the origin of r_2.

2. Printed schedules for each trip ρ_r and the mapping $\mathbb{M}_r$ are assumed to be known for all $r \in R$.

3. All buses have infinite capacity.

4. Travel times on all arcs are assumed to be integer-valued.

5. The time taken by buses during boarding and egress of passengers is neglected.

6. Buses begin and end trips/runs at a garage, i.e., a bus in the network is always assumed to serve a route. This assumption is not restrictive as a bus that goes out of the network and comes back at a later point of time may be considered as a new bus.

7. Travel times on transit arcs are independent random variables with known probability distributions with finite support. It is also assumed that the lowest possible travel time realization on a transit arc (i, j) is the difference of the scheduled arrival times at j and i. Though the assumption seems restrictive, it is reasonable in practice as drivers in several cities are instructed to wait at bus

stops if they are ahead of schedule. Higher travel time realizations of a transit arc are assumed to occur under congested conditions.

8. We assume without loss of generality that travel time pmfs on transit arcs vary only across trips of a route. However, the subject developed in this thesis can be applied to cases in which pmfs vary with time.

9. The first trip made by a bus starts on time and buses begin new trips on schedule if they can, by adjusting the slack provided in schedules between trips. Hence, if a bus arrives at or after the scheduled departure time at the origin of the next trip, the bus is assumed to proceed along the next trip immediately.

10. Bus bunching and overtaking is permitted, i.e., FIFO order need not be preserved.

Some of the assumptions made above can be found in existing literature. For instance, assumptions 3 and 5 and are similar to the ones used by Hickman [13] and, Hickman and Bernstein [15]. While these authors assume that the slack in the schedules is utilized even if a bus is delayed, we use assumption 9 instead as it is more practical. Assumptions 4, 7 and 8 are borrowed from Miller-Hooks [18] and, Miller-Hooks and Mahmassani [19].

3.3 An Example

Consider the following example to illustrate the notation used and the network transformations described in the previous sections. Suppose, two routes 1 EB/WB and 2 NB/SB serve stops shown in the figure 3.1. The arcs in the figure includes all transit arcs (A_r). As mentioned earlier, routes in opposite directions are modeled as different routes. If at least one bus serves two routes in succession and the destination of first is the origin of the second, then the common node is replicated to model the slack (for e.g. bus stop 3).

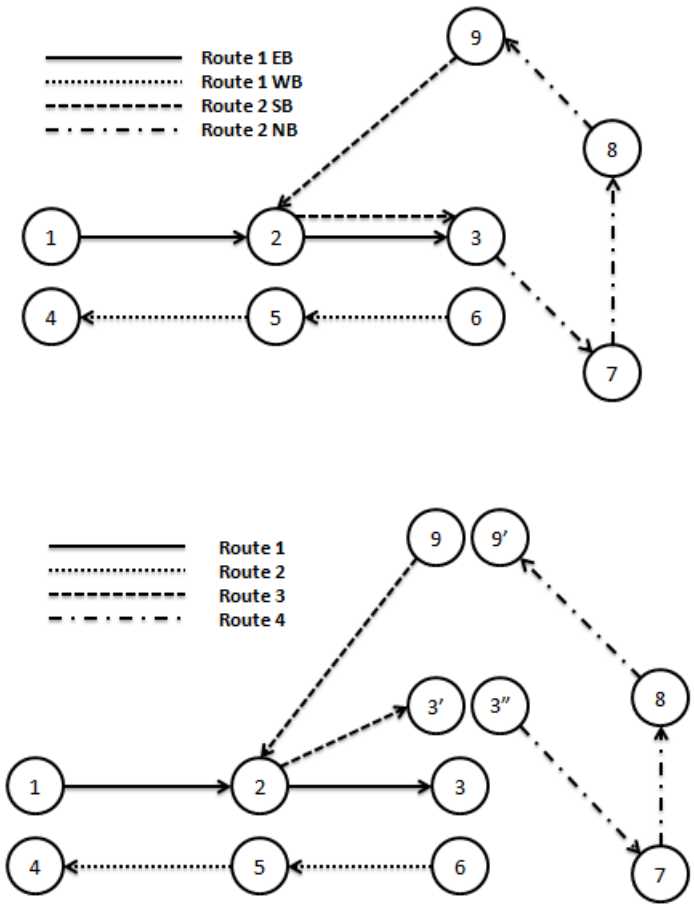

Figure 3.1: Physical transit network(top) and transformed network(bottom)

Tables 3.2 and 3.3 shows the schedules (which may be obtained from the local transit agency) of trips on routes and the buses serving them. The values in the tables indicate the arrival time of buses from the instant at which the first bus enters the network.

Table 3.2: Schedules on routes 1 EB and 1 WB

Route r_1					*Route* r_2				
		Stop					*Stop*		
ρ_{r_1}	$\mathbb{M}_{r_1}(\rho_{r_1})$	1	2	3	ρ_{r_2}	$\mathbb{M}_{r_2}(\rho_{r_2})$	6	5	4
$\rho^1_{r_1}$	b_1	0	12	20	$\rho^1_{r_2}$	b_3	30	44	58
$\rho^2_{r_1}$	b_2	25	38	54	$\rho^2_{r_2}$	b_2	60	72	80
$\rho^3_{r_1}$	b_3	60	75	90	$\rho^3_{r_2}$	b_1	80	96	108

Table 3.3: Schedules on routes 2 NB and 2 SB

Route r_3					*Route* r_4					
		Stop					*Stop*			
ρ_{r_3}	$\mathbb{M}_{r_3}(\rho_{r_3})$	9	2	3	ρ_{r_4}	$\mathbb{M}_{r_4}(\rho_{r_4})$	3	7	8	9
$\rho^1_{r_3}$	b_4	16	28	40	$\rho^1_{r_4}$	b_1	24	36	40	44
$\rho^2_{r_3}$	b_1	50	60	74	$\rho^2_{r_4}$	b_4	48	64	70	76

Using this data, the itineraries of a bus in the network are created as shown in table 3.4, which are then used to construct the individual states of the bus. Although the notation employed is a bit cumbersome, it avoids ambiguity between the trips of a bus and the trips on a route. But, when it is clear from the context, we suppress the subscripts and simply refer to buses and states by arabic numerals (i.e., a bus is written as 1 as opposed to b_1).

As mentioned earlier, we use N_{β_b} and A_{β_b} to denote the nodes and arcs visited by a bus while serving a particular trip. For instance in the following example, $N_{\beta^2_{b_1}} = \{3'', 7', 8, 9'\}$ and $A_{\beta^2_{b_1}} = \{(3'', 7), (7', 8), (8, 9')\}$.

Table 3.4: Itinerary of bus I_{b_1}

$\beta_{b_1} \rightarrow$	$\beta^1_{b_1}$			$\beta^2_{b_1}$				$\beta^3_{b_1}$			$\beta^4_{b_1}$		
$n_{\beta_{b_1}}$	1	2	3	3”	7 ’	8	9’	9	2	3’	6	5	4
$f_{n_{\beta_{b_1}}}$	0	12	20	24	36	40	44	50	60	74	80	96	108

3.4 Constructing individual states of a bus

In order to construct the individual states of a bus, we make use of the current state vector, the set of trips I_b and the pmfs on arcs for a particular trip $\Omega_{a_{\beta_b}}$. Note that when a traveler departs at a node in the network, buses might either be present in the network or at the garage. By convention, we assume that the state of a bus that has not yet been in the network is represented by the node-time pair $(0, -1)$ and a bus in the garage that left the network is characterized by the state $(0, -2)$.

If a bus is present in the network, the current state vector provides information about the most recently visited stop and the time at which the bus visited it. This gives us the first state of the bus and the states at subsequent nodes are obtained using the travel time distributions. We assume that the travel time distributions of an arc traversed by a bus on a particular trip remains the same for every possible arrival time at its tail node. When route changes occur, the arc that connects the destination of one route and the origin of the other is modeled to reflect the assumptions made about the slack in schedules (see assumption 9). If successive routes served by a bus are such that the destination of the first route is the same as the origin of the next route, by construction we have a copy of the node for each route and the link connecting these nodes is used to model the slack. The following figure illustrates the process of construction of individual states of a bus, consistent with the assumed pmfs. Note that the link travel times on arcs representing the slack depends on the state and not on the trip of a bus.

Consider a bus b, which was scheduled to arrive at stop 1 at time $t = 2$ (see figure 3.2). Suppose, the current system state vector indicates that the bus was last seen at stop 1 at time $t = 3$. Since the travel time on the first link is either 4 or 8, the possible arrival time at the next stop is either 7 or 11. Proceeding in a similar manner, we find the states at node 3, for each possible arrival time at node 2. Let node 3 be the

destination node of route r_1 and the origin node of route r_2. For the sake of brevity, we drop the indices of $t_{n_{\beta_b}}$ and let t_3 represent the arrival time of the bus at node 3. In order to model the slack, the cost of arc $(3, 3')$ is defined as $(15 - t_3)^+$. This construct ensures that if the bus arrives at node 3 at $t_3 = 11$ or 13, it waits for 4 or 2 min respectively, but if it arrives at $t_3 = 15$ or 18, it proceeds immediately to serve the next route. Starting with the initial state, the states of a bus are assumed to spatially ordered based on the number of nodes visited by the bus along its trips. For instance, $order((1, 3)) = 1$ as it is the first node visited by the bus from the current state. Likewise, order of states (2,7) and (2,11) is 2 and so on.

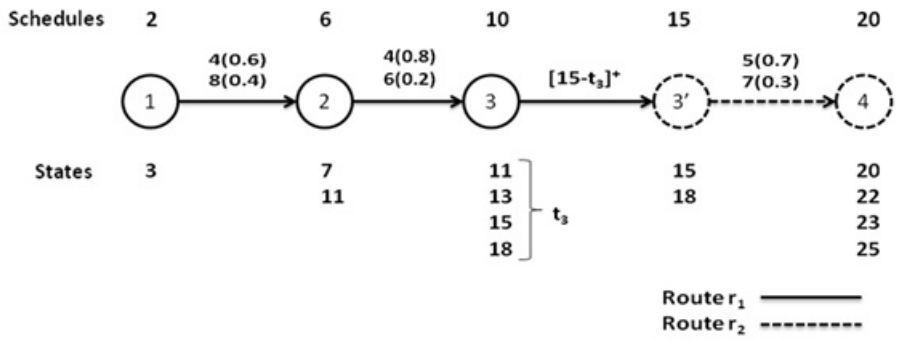

Figure 3.2: Individual states of a bus

If a bus is found to be present in the garage in the current state, we append a node 0 (used to represent the garage) before the start of its first trip. For example, suppose a traveler departs from his/her origin node at $t = 3$ and a bus b is scheduled to begin its trips from node 1 at $t = 14$. Since the bus is not present in the network at $t = 3$, we add a node 0 and an arc (0,1) as shown in figure 3.3. This network transformation is necessary for the construction of the system state space. Alternatively, we could model this by assuming that the bus is present at node 0 at times t = 3,.....14 and let the arc cost on arc (0,1) be equal to $14 - t_0$, where t_0 is the time at which the bus is found at node 0. However, this increases the number of states which can be avoided using the earlier method (the details of which are discussed later).

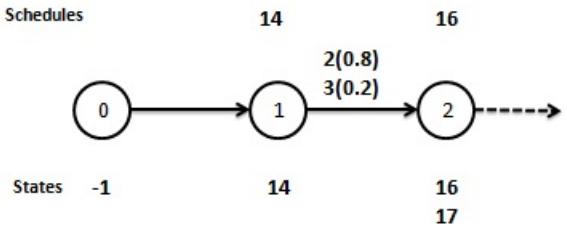

Figure 3.3: Individual States of a bus present in the garage

3.5 Formal definition of the ATR problem

Unlike in regular traffic networks, a traveler in a public transportation system can choose routes that minimize travel time, fares, number of transfers, walking distance or take into consideration available capacity on buses. However, with regard to adaptive routing in stochastic networks, these objectives are themselves random variables. Although, it is commonly assumed that the choice of a traveler is purely governed by minimization of the expected cost of travel, such an objective might lead to unfavorable outcomes with a small probability. For instance, in minimizing the expected travel time, a traveler could potentially deviate from his/her path significantly, or cycle several times (note that since we have a finite horizon it is not possible to cycle indefinitely) before reaching the destination. Thus, travelers could be deterred from the risks involved in using the expected value as the sole criterion for choosing routes. In such situations, other objectives such as minimization of the worst possible outcome or the weighted sum of expected value and the variance etc. are usually used to model the risk averse attitude of decision makers.

The objective and the measure of risk aversion for the ATR problem is defined as follows. Let X be a random variable that denotes the cost of the adaptive strategy. We wish to minimize the expected value of X such that $\mathbb{P}(t_O + X \leq \mathit{cutoff}) = 1$, for some given value of *cutoff*. In our experience, trying to find a least expected cost path

with a high probability (instead of 1) of arriving at the destination before a given time or threshold is harder to solve using the state space reduction techniques developed in this thesis. Although we might find a policy that results in a better objective function value in the absence of such a constraint, incorporating these behavioral measures makes it more practical and further helps in the reduction of the state space.

Based on the above discussion, we now formally define the ATR problem. Given a stochastic transit network (in which the link travel times are time dependent and random with known discrete distributions), the initial state of the system and a destination, we intend to find an optimal policy conditional on the current state of the system that minimizes the total expected travel time subject to a measure of risk aversion. The problem can be easily extended to include other modes of transit such as rail by including the set of the train lines and trains in the set of routes and buses respectively.

Clearly, the number of individual states grows exponentially with increase in the number of trips made by the bus, which in turn results in an exponential number of system states and thus the ATR problem exhibits *the curse of dimensionality*. Solving the ATR problem as an MDP by application of the Bellman's principle or other approximation techniques thus becomes extremely difficult unless we find ways to reduce the size of the state space, which is discussed in the next chapter.

Chapter 4

Preprocessing

Using the individual states of buses, shortest path algorithms are developed in this chapter, the advantages of which are twofold. First, they provide a rationale for defining the value of the *cutoff* which characterizes the risk averse attitude of a traveler. Secondly, they help in the development of the elimination procedure which is discussed in later sections. In the last section, the elimination methods are compared to the principle of light cones, a concept used in physics to describe causality.

4.1 Algorithms for preprocessing

The algorithms used for the preprocessing methods comprise of variants of the TDSP problem. In this section, we define and solve these algorithms using labeling approaches. More specifically, the following sets of problems are considered.

(a) *Earliest Origin-to-All TDSP (EOA):*
Given the departure time at the origin, the EOA problem involves computation of shortest path labels, where the label of a node represents the earliest possible time at which we can reach the node with positive probability ($w.p. > 0$).

(b) *Earliest All-to-Destination TDSP (EAD):*
This involves finding shortest path labels which specify the earliest we can reach the destination from all nodes $w.p. > 0$, for all possible departure times.

(c) *Latest Origin-to-All TDSP (LOA):*
In this problem, given the departure time at the origin, we find labels which represent the earliest we can reach a node with probability 1 ($wp1$).

(d) *Latest All-to-Destination TDSP (LAD):*

Given an individual state of a bus, the LAD problem determines the earliest we can reach the destination $wp1$, assuming that the individual state of the traveler is same as that of the bus. The *all* in LAD refers to individual states and should not be confused with nodes in the network.

In order to provide a road map for the subject developed in this chapter let us briefly review the use of these algorithms before discussing them in detail. The first step in the preprocessing procedure involves defining the risk averse measure or the *cutoff*. As different individuals are known to have different risk attitudes, the value of *cutoff* for each traveler could be different, and in some cases difficult to define. For travelers with hard constraints (for e.g. the traveler has to board a flight) the arrival time can be used as the *cutoff*. In other cases, one might think along the lines of guaranteed returns to define the value of *cutoff* i.e., if a traveler is guaranteed a strategy that ensures that the destination is reached within *cutoff*, he/she might not be inclined to follow a strategy which could potentially take longer to reach the destination. A naive *cutoff* that can be used is the shortest walking time between the origin and the destination. Although finding this is easy, it might still be very high for long trips and is thus impractical. In this thesis, a *cutoff* defined by the LOA label to the destination, which represents the earliest time by which we are guaranteed to reach the destination is used. Note that this value of the *cutoff* also ensures that there exists at least one feasible solution to the ATR problem.

After fixing the value of the *cutoff*, an elimination procedure is developed to discard states of buses that do not affect the optimal strategy. Since the traveler has to reach the destination within a threshold $wp1$, the above labels (particularly the EAD and LAD labels) may be used to verify if boarding a bus at a specific individual state violates the risk aversion constraint. Additionally, we use the EOA labels to check if we can catch a bus at an individual state. The conditions under which an individual

state can be ignored are then developed which is examined in detail in later sections.

Let us now turn our attention to the algorithms for computing these labels. Let O and D be the origin and destination node respectively and let SE represent a scan eligible list. In addition, let $\Gamma(i)$ denote the set of nodes adjacent to node i and let $\Gamma^{-1}(i)$ denote the set of nodes from which we can reach node i directly (i.e., $j \in \Gamma^{-1}(i) \Leftrightarrow (j,i) \in A$).

4.1.1 Earliest Origin-to-All TDSP (EOA)

In this section, a label correcting algorithm for computing the EOA labels is presented. Suppose that μ_n represents the EOA label of a node n, where $n \in N$. In order to find the earliest time by which a node in the network can be reached $w.p. > 0$, we define a time dependent cost parameter $\zeta_{ij}(t)\ \forall\ t \geq t_O$ as follows:

$$\zeta_{ij}(t) = \min \left[\min_{\substack{b \in B,\ \beta_b \in I_b,\ a_{\beta_b} \in A_{\beta_b} \\ :\ t \in T_{i_{\beta_b}} \&\ a_{\beta_b} = (i,j)}} C^{[1]}_{a_{\beta_b}},\ w_{ij} \right]$$

The cost of arc (i,j) for a departure time t is set to the minimum of the walking travel time between i and j and the lowest possible transit cost on the arc (i,j) among all buses that reach node i $w.p. > 0$ at time t.

Consider the network shown in figure 4.1 to illustrate the EOA problem. Suppose two buses b_1 and b_2 start at nodes 1 and 4 at $t = 0$ on routes r_1 and r_2 respectively. Let the cost of transit arcs be as shown in the figure and let the walking travel times on these arcs be 40. Assume a traveler headed to node 5 departs from node 1(origin) at $t = 0$.

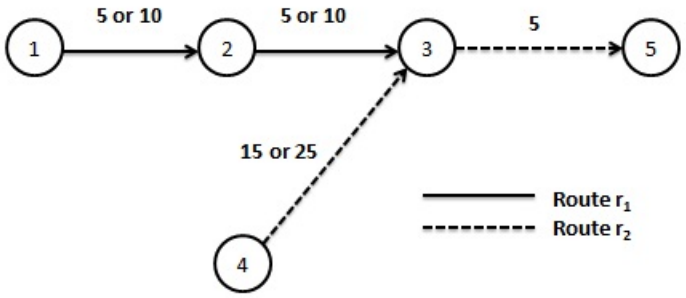

Figure 4.1: Network to illustrate the preprocessing algorithms

The earliest a traveler can reach the destination is 20, provided bus b_1 takes 5 minutes to travel on arcs (1,2) and (2,3) and bus b_2 takes 15 minutes to travel on arc (4,3). It is easy to see that the proposed algorithm, achieves this result by solving a TDSP with time dependent arc costs as described earlier.

Algorithm 1 *Pseudocode for EOA*

Step 0: Initialize Labels
$\mu_O = t_O$
$\mu_n = \infty \ \forall \ n \in N \backslash \{O\}$
SE $= \{O\}$
$Pred(O) = 0$

Step 1:
while SE $\neq \varnothing$ **do**
  Remove a node i from SE
  for each $j \in \Gamma(i)$ **do**
    for all $t \geq \mu_i$ **do**
      if $\mu_j > t + \zeta_{ij}(t)$ **then**
        $\mu_j = t + \zeta_{ij}(t)$
        $Pred(j) = i$
        If $j \notin$ SE add j
      end if
    end for
  end for
end while

The pseudocode described above is a label correcting algorithm for TDSP with waiting allowed, in which the time dependent link costs are defined by $\zeta_{ij}(t)$. Starting with the origin, the algorithm scans downstream nodes and updates their labels if the optimality condition is not satisfied. Nodes whose labels change are added to the SE list and are examined later. The algorithm also keeps track of predecessor labels which can be used to construct the optimal path to a node.

Observe that the EOA labels cannot be calculated by assuming that all buses in the network travel at their fastest pace. For instance, if the cost of the arc (4,3) was 5 or 10 and if buses take the lowest possible time to traverse arcs, the traveler would miss bus b_2. However, the traveler might get lucky if bus b_2 is delayed, in which case he/she can reach the destination at $t = 15$.

4.1.2 Earliest All-to-Destination TDSP (EAD)

The EAD problem is a straight forward extension of the EOA problem. The time dependent arc costs $\zeta_{ij}(t)$ defined earlier are used to compute labels $\gamma_n(t)$, which represent the earliest we can reach the destination $w.p. > 0$, given that we depart from node n at time t.

The algorithm first adds the destination to the scan eligible list and updates the labels of the upstream nodes in order to satisfy the optimality criterion. Successor labels are modified when updates occur and can be used to construct the optimal path. We assume $\forall\ t \geq T,\ \gamma_n(t) = \gamma_n(T-1)$, which is reflective of a steady state (i.e., arc costs are no longer time dependent) after time T. This assumption is not restrictive as long as T is sufficiently large, in which case the arc costs equal the walking travel times. Given below is the routine for estimating the EAD labels which is similar to an all-to-one TDSP algorithm with waiting allowed.

Algorithm 2 *Pseudocode for EAD*

Step 0: Initialize Labels
$\gamma_D(t) = 0 \; \forall \; t \in T$
$\gamma_n(t) = \infty \; \forall \; n \in N \backslash \{D\}, \; t \in T$
SE $= \{D\}$
$Succ_D(t) = 0 \; \forall \; t \in T$

Step 1:
while SE $\neq \varnothing$ **do**
  Remove a node j from SE
  for each $i \in \Gamma^{-1}(j)$ **do**
    for all $t \geq t_O$ **do**
      if $\gamma_i(t) > \zeta_{ij}(t) + \gamma_j(t + \zeta_{ij}(t))$ **then**
        $\gamma_i(t) = \zeta_{ij}(t) + \gamma_j(t + \zeta_{ij}(t)))$
        $Succ_i(t) = j$
        If $i \notin$ SE add i
      end if
    end for
  end for
end while

4.1.3 Latest Origin-to-all TDSP (LOA)

This problem involves finding the earliest time by which we are guaranteed to reach a node, given the departure time at the origin. Let us denote the LOA labels by λ_n, where $n \in N$. To compute these labels, a modified label setting method is applied that ensures all transfers are made $wp1$ by checking if the *latest arrival time at a transfer or boarding node is lesser than or equal to the earliest possible departure time of a bus serving that node.* Consider the network shown in figure 4.1 on page 27. Clearly, it may be inferred from the figure that the traveler can reach node 3 at $t = 20$ $wp1$. However, the traveler might miss bus b_2 (if it arrives at 3 at $t = 15$) and hence the LOA label of node 5 is 20+40=60. Note that the LOA labels cannot be computed using a TDSP on a network with the longest possible travel time on the transit arcs. Fixing the travel time on transit arcs to the largest possible values incorrectly updates the label λ_1 to 30, as we can catch the second bus after reaching

node 3 at $t = 20$.

Consider the following approach. Starting from the origin let us scan the downstream arcs (assuming waiting is allowed) and if a particular bus can be caught $wp1$, we use a time dependent arc cost which is the minimum of the walking travel time and the largest possible transit arc cost, experienced by a bus, assuming that it arrives at its last possible state (temporally). We then check for the optimality of labels of the downstream nodes and proceed till the algorithm terminates. However, this approach may fail in some cases as demonstrated in the network in figure 4.2.

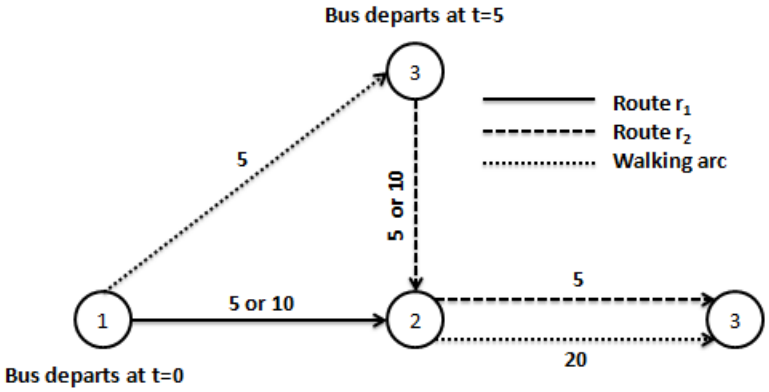

Figure 4.2: Application of the LOA algorithm

Let two buses b_1 and b_2 start on route r_1 and r_2 at $t = 0$ and $t = 5$ respectively. Assume the LOA labels are computed for a traveler departing at $t = 0$ from node 1. Following the procedure described earlier, the label of node 2 is first updated to 10. Upon examination of arc (2,3), the algorithm concludes that b_2 cannot be caught with $wp1$ and hence, the label of 3 is modified to 30 using the walking arc (2,3). The problem with this approach lies in the fact that sub-path optimality does not hold. The traveler can board b_2 after walking to node 3 $wp1$ and hence reach the destination at $t = 15$ $wp1$. Therefore, when an arc is used to update the label of a downstream node, it is also important to check if the a bus that traverses the arc can be boarded at some other node in the network $wp1$.

In order to account for such cases, we define a variables φ_b, which is set to 1 if a bus b can be boarded $wp1$ and φ'_b, which represents the first stop among the stops in its itinerary at which it can be boarded $wp1$. When a node is examined, we update the variable φ_b to 1 if a bus traversing one of its downstream arcs can be boarded $wp1$. We then check for the optimality of labels of the downstream nodes and if a transit arc served by b is used, we make sure φ_b is 1 before updating the label. Let us now apply this method to the network in figure 4.2. When node 3 is observed, we update φ_{b_2} to 1 as we know that b_2 can be surely boarded. Thus, when 2 is examined, the transit arc (2,3) can be used to update the label of node 3. However, the validity of this method is still questionable as examining node 2 before 3 will not guarantee optimal labels.

Now suppose nodes are scanned in increasing order of labels. While this modification ensures that 3 is scanned before 2, it may fail if the labels of nodes 2 and 3 are tied. Breaking ties arbitrarily can result in incorrect labels as one may examine node 2 ahead of 3. Therefore, we scan all downstream arcs of all nodes with the minimum label to update φ_b after which we select a node, remove it from the scan eligible list and check for the optimality criteria.

We first define parameters $\varsigma_{ij}^b(t)$ as shown below. $\varsigma_{ij}^b(t)\ \forall\ t \geq t_O$ is set to the largest possible travel time on arc (i,j) when a bus b on a particular trip is at its last possible individual state at node i.

$$\varsigma_{ij}^b(t) = \begin{cases} C_{a_{\beta_b}}^{|\Omega_{a_{\beta_b}}|} & \text{if } \exists\ \beta_b \in I_b,\ a_{\beta_b} \in A_{\beta_b} :\ a_{\beta_b} = (i,j)\ \&\ t = l_{i_{\beta_b}} \\ \infty & \text{otherwise} \end{cases}$$

Also, assume $\Phi_{n,t} \in B$ is a subset of buses that can be caught $wp1$ at node n at a time $\geq t$ and let $\Phi'_{n,t}(b)$ denote the order of the earliest possible state at which a bus

$b \in \Phi_{n,t}$ can be boarded at node n $wp1$. An indicator variable ϵ_i is used to check if node i has been previously examined to update the φ values. Using these parameters and definitions, we hypothesize that the following algorithm computes the optimal LOA labels. However, a proof of correctness has not yet been established.

Algorithm 3 *Pseudocode For LOA*

Step 0: Initialize Labels
$\lambda_O = t_O$
$\lambda_n = \infty \ \forall \ n \in N \backslash \{O\}$
SE $= \{O\}$
$\varphi_b = 0 \ \forall \ b \in B$
$\varphi'_b = \infty \ \forall \ b \in B$
$\epsilon_n = 0 \ \forall \ n \in N$

Step 1:
while SE $\neq \varnothing$ **do**
  for all $i : \lambda_i = \min\{\lambda_k : k \in \text{SE}\}$ & $\epsilon_i \neq 1$ **do** ▷ Step 1.1
    for all $b \in \Phi_{i,\lambda_i}$ **do**
      $\varphi_b = 1$
      $\varphi'_b = \min\{\varphi'_b,\ \Phi'_{i,\lambda_i}(b)\}$
    end for
    $\epsilon_i = 1$
  end for

  Remove a node $i : \lambda_i = \min\{\lambda_k : k \in \text{SE}\}$ ▷ Step 1.2
  for each $j \in \Gamma(i)$ **do**
    for all $t \geq \lambda_i$ **do**
      $d_{ij}(t) = w_{ij}$
      for all $b \in B$, $\beta_b \in I_b$, $i_{\beta_b} \in N_{\beta_b} : t = l_{i_{\beta_b}}$ & $\varphi'_b \leq order_b((i,t))$ **do**
        $d_{ij}(t) = \min\{\varsigma^b_{ij}(t),\ w_{ij}\}$
      end for

      if $\lambda_j > t + d_{ij}(t)$ **then**
        $\lambda_j = t + d_{ij}(t)$
        If $j \notin$ SE add j
      end if

    end for
  end for
end while

In step 1.1, the algorithm updates the information of buses that can be caught $wp1$ from all nodes which have the least label and in step 1.2, it picks a node with the smallest label and scans all of its adjacency list and updates their labels using the walking travel time or the largest possible travel time on transit arcs served by buses that can be boarded $wp1$. In the above algorithm, $order(s_b)$ is indexed by buses as $order_b((i,t))$ to avoid ambiguity when two or more buses have the same state.

Alternately, the LOA labels can be computed using a recursive application of a TDSP algorithm by varying the time dependent arc costs. We first assume that the time dependent arc costs are same as the shortest walking arc travel times and solve a TDSP to obtain the earliest time by which we can reach a node in the network. Using these labels, we find the first stop at which a bus in the network can be boarded $wp1$. The time dependent arc costs are then updated assuming that the bus takes the longest possible time to traverse links on trips beyond the stop at which it could be boarded $wp1$. The TDSP labels are computed again and this process is repeated until the labels do not change.

4.1.4 Latest All-to-Destination TDSP (LAD)

The LAD problem finds the earliest we can reach the destination $wp1$, for each individual state of a bus, assuming that the state of a traveler is same as that of the bus. Let $\eta(s_b)$ denote the LAD labels, where s_b represents the individual state of a bus b. To compute these labels, we make minor modifications to the input parameters of the LOA algorithm and solve it repeatedly for each individual state.

Let us revisit the example shown in figure 4.1 on page 27. The individual states and LAD labels of each state for both buses are shown in the figure 4.3. Consider the bus on the first route. Assume the state of the bus is (2,5), i.e. it is at node 2 at $t = 5$. As the traveler can board bus b_1, he/she can reach node 3 at $t = 15$ $wp1$, and can

successfully transfer to bus b_2 and reach the destination at $t = 30$ $wp1$.

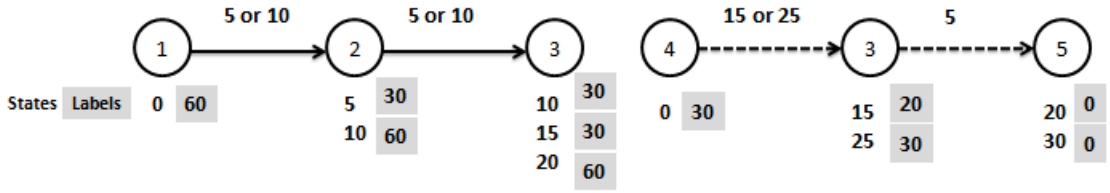

Figure 4.3: LAD labels of individual states of bus b_1(left) and b_2(right)

Since, the traveler's individual state is same as that of the bus b, he/she can board bus b $wp1$. Therefore, in the initialization step of the algorithm we set $\varphi_b = 1$ and $\varphi'_b = order(s_b)$. Note that the origin of the traveler is $\tilde{n}(s_b)$ and hence we set $\lambda_{\tilde{n}(s_b)} = \tilde{t}(s_b)$. The LOA algorithm may then be used to calculate the label of the destination (which is also the LAD label for state s_b).

4.2 Preprocessing procedure

The optimal strategy of the ATR problem is not necessarily influenced by all buses in the network. For instance, while traveling between an OD pair, buses that ply on routes far away from the OD pair may not have an impact the optimal policy. Further, as the trips made by buses beyond a certain point of time are not relevant, only a limited number of individual states are useful in solving the problem. The preprocessing steps described in this section aims at identifying the buses and individual states that could affect travel between an OD pair using the concept of risk aversion and the algorithms described earlier and significantly reduces the size of the state space. Given below is a summary of the preprocessing procedure followed by a detailed discussion of the steps involved.

1. Solve for the LOA labels and obtain the value of λ_D
2. Perform initial elimination of individual states
3. Redefine the size of T - the time period of interest

4. Solve for EOA, EAD and LAD labels
5. Define indicator variables using these labels, eliminate states and create absorbing/sink states

Solve for the LOA labels:

In this step we use algorithm 3 to compute λ_D, the earliest time by which we can reach the destination $wp1$. This is used to define the *cutoff* as explained earlier. Clearly, the size of state space reduces with decrease in the value of the *cutoff*, but the methods for preprocessing described in later sections do not depend on the value used.

Perform initial elimination:

In order to ensure $\mathbb{P}(X+t_O \leq \lambda_D) = 1$, we discard all individual states of a bus beyond a certain point. More precisely, if the earliest time at which a bus reaches a node is greater than λ_D, we eliminate all individual states associated with the remainder of its journey. Although this step is not necessary, it speeds up the estimation of other labels as the complexity involved in their computation is a function of $|T|$.

Redefine T - time period of interest:

We can now reduce the time period of interest T to a much smaller set. However, at this stage we may have cases in which buses reaches individual states at $t > \lambda_D$ and hence we can't use $|T| - 1 = \lambda_D$. Instead, we choose $|T|$ to be sufficiently large such that it is greater than or equal to the time corresponding to the individual state of any bus in the network.

Solve for EOA, EAD and LAD labels:

In this step we use algorithms 1, 2 and 3, to find the EOA, EAD and LAD labels respectively.

Define indicator variables, eliminate states and create absorbing states:

The final stage of the preprocessing procedure aids in the elimination of extraneous

individual states and buses. The ideas behind this step are primarily motivated by the following questions:

1. *Suppose the destination of a traveler is to the south of the origin node. Should he/she consider the states of a bus heading in the opposite direction (i.e., north bound)? If yes, do all the individual states of the bus play a role in finding the optimal strategy?*
2. *Suppose a traveler missed a bus. Can we exclude it from the set of buses to be considered while populating the system states?*
3. *Buses b_1 and b_2 have the same itinerary and are such that b_1 is ahead of b_2. If a traveler at a bus stop can board bus b_1, can the individual states of bus b_2 be ignored?*

While the first two questions prompt us to utilize the labels computed in step 4 to eliminate individual states, the third question helps us understand the limitations of using a dominance argument. Let us now examine the elimination process in detail.

4.2.1 Elimination based on EAD/LAD labels

Consider the first question. A traveler heading south might reach the destination faster by transferring to another bus after boarding the north bound bus, thus making a case for its inclusion in constructing the system state space. In other words, a traveler may travel away from the destination geographically, in the process of minimizing the travel time.

Let us now address the second part of the question. We might be able to limit the individual states of a bus by discarding the ones from which we cannot make it to the destination before the *cutoff*. This is accomplished using the EAD or LAD labels by defining indicator variables δ_{ead} and δ_{lad} as shown below.

$$\delta_{lad}(s_b) = \begin{cases} 1 & \text{if } \eta(s_b) > \lambda_D \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{ead}(s_b) = \begin{cases} 1 & \text{if } \gamma_{\tilde{n}(s_b)}(\tilde{t}(s_b)) > \lambda_D \\ 0 & \text{otherwise} \end{cases}$$

It is obvious that $\delta_{ead}(s_b) = 1 \Rightarrow \delta_{lad}(s_b) = 1$ and $\delta_{lad}(s_b) = 0 \Rightarrow \delta_{ead}(s_b) = 0$. Let us analyze the case where $\delta_{lad}(s_b) = 1$ and $\delta_{ead}(s_b) = 0$. The value of $\delta_{lad}(s_b)$ implies that by boarding the bus at the individual state s_b we cannot reach the destination within *cutoff* $wp1$, but for some particular state/states of the system the destination can be reached before the *cutoff*. Under such circumstances, based on the system state, the algorithm might direct the traveler to catch the bus if it foresees that taking the bus does not violate the risk aversion condition and might provide a different strategy otherwise. Hence, by using LAD labels one might eliminate more states than required. Although we study the properties and results of elimination based on both, EAD and LAD labels, it is beyond the scope of this thesis to explore the trade-off between the computational advantages of dealing with smaller state space (obtained by using elimination based on LAD labels) and the value of the objective.

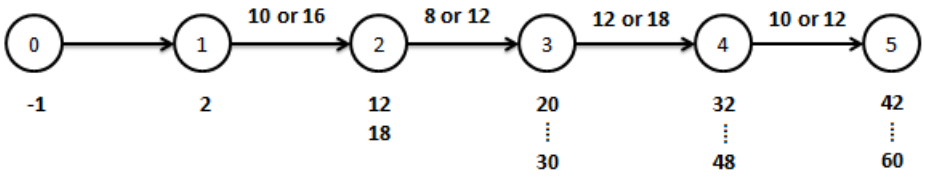

Figure 4.4: Elimination of the individual states of a bus

Let us study the elimination process using the example shown in figure 4.4. Table 4.1 contains the delta values in addition to the list of individual states of the bus (excluding the garage state). Assume that the *cutoff* obtained by solving the LOA algorithm

is 40. Therefore, states beyond node 4 can be discarded according to the second step of the preprocessing procedure.

Table 4.1: δ_{ead} and δ_{lad} values for individual states of the bus

State(s_b)	Node($\tilde{n}(s_b)$)	Time($\tilde{t}(s_b)$)	δ_{ead}	δ_{lad}
1	1	2	0	1
2	2	12	0	1
3	2	18	1	1
4	3	20	0	1
5	3	24	0	1
6	3	26	1	1
7	3	30	1	1
8	4	32	0	0
9	4	36	0	1
10	4	38	1	1
11	3	42	1	1
12	4	44	1	1
13	4	48	1	1

We first discuss the elimination methods based on the EAD labels and later extend it to the LAD labels. Let the set of individual states of a bus be represented as an acyclic network as shown in figure 4.5. A node in the network denotes an individual state and arcs are used to connect adjacent states. Let all states that can be reached from an individual state and all states from which a particular state can be reached be referred to as *descendant* and *ancestor* states respectively.

For the ease of illustration a few states have been replicated (indicated by the dashed connectors). All states with $\delta_{ead} = 1$ are marked in gray. Note that if for a particular state $\delta_{ead} = 1$, we cannot reach the destination within the *cutoff* with positive probability from all descendant states. The elimination of individual states can be divided into the following two phases.

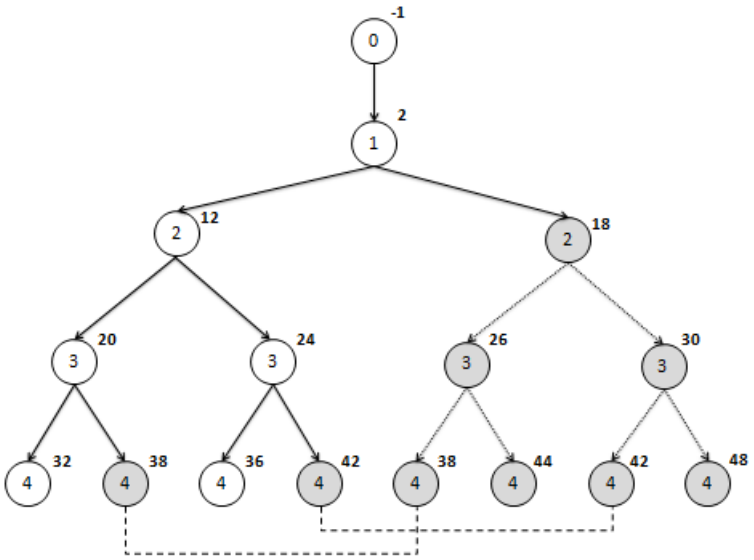

Figure 4.5: Acyclic network structure of states marked with w.p. > 0 labels

Phase I:

From figure 4.5, observe that if a bus at state (1,2) takes 16 minutes to travel to node 2, we can be certain that it does not influence the optimal strategy. Thus if the travel time between nodes 1 and 2 is 16, we assume the bus exits the network. This is achieved by creating an absorbing/sink state $(0, -2)$ which indicates that the bus is back in the garage. The phase I elimination procedure can be described as follows. In the acyclic network of individual states, delete arcs that connect marked states (shown by the dotted lines in figure 4.5) and then delete all marked states. Arcs orphaned after the removal of marked states are then connected to the garage state $(0, -2)$. If multiple arcs are created between an individual state and the sink state, replace them with a single arc between the two states. Finally, connect all unmarked states (excluding the garage states) with outdegree 0 to the state $(0, -2)$. Buses from these states are assumed to incur a cost of 0 to reach the sink. The resulting network of individual states is shown to the left of the following figure.

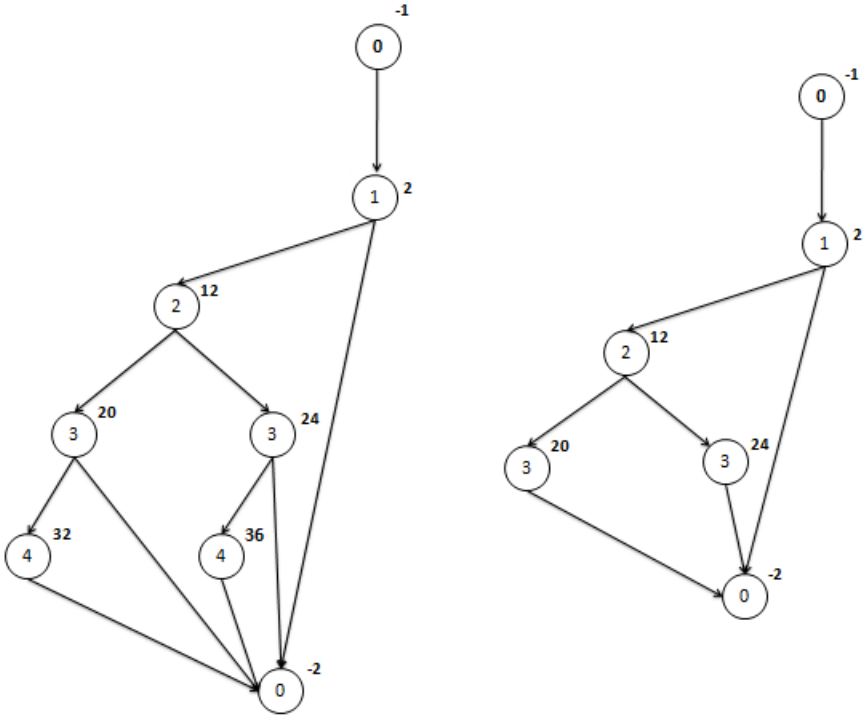

Figure 4.6: Phase I(left) and Phase II(right) elimination of individual states

Phase II:

Suppose a traveler is about to board the bus at node 3 at $t = 20$. Although the destination can be reached within the *cutoff* with positive probability, the risk aversion condition may be violated if the bus takes 18 minutes to travel on its next arc. Hence, the state (4,32) can be eliminated and the bus may be assumed to proceed immediately to the garage from state (3,20). The state (4,36) can also be eliminated using a similar argument.

In general, assuming the state with the highest order is denoted by y_{max} we proceed as follows. If y_{max} is 1 the bus is completely ignored. Else, pick individual states with order y_{max}. If at least one of the outgoing arcs from each predecessor state of the state with order y_{max} is connected to the garage state $(0, -2)$, we delete the state

being examined and the orphaned arcs. If no state is eliminated we terminate, else y_{max} is recalculated and the procedure is repeated.

So far we have discussed the use of EAD labels to eliminate individual states. In a similar vein, a stronger but approximate elimination method using the LAD labels can be developed. We begin by marking states using the δ_{lad} values, but unlike before, a state is marked if at least one of its descendants is marked. Further, we unmark all marked ancestor states of unmarked states to prevent loss of information. Finally, we employ phase I and phase II techniques to eliminate individual states. The following figure shows the acyclic network in which states have been marked using δ_{lad} values and the resulting network from phase I. As explained earlier, we unmark states (3,20), (2,12) and (1,2) before beginning phase II as the state (4,32) is unmarked. Upon using phase II, all states can be eliminated, thus leading to the exclusion of the bus while populating the system state space.

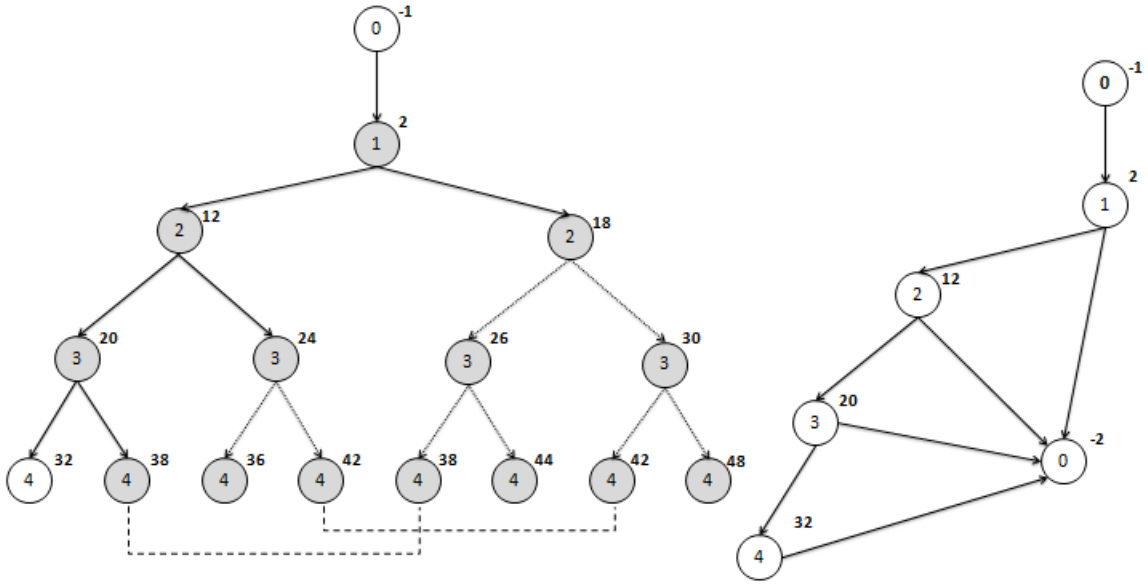

Figure 4.7: Acyclic network structure of states marked with wp1 labels(left) and the modified individual states from phase I(right)

Effects of elimination on the labels and delta values:

The elimination of individual states of each bus was carried out independently without regard to its impact on the labels. It is thus necessary to check if the elimination changes the labels to an extent that it compromises the validity of the entire procedure. Suppose we eliminate states based on EAD labels. In phase I, if a particular state is removed, we know that by boarding the bus at that state we cannot reach the destination within the *cutoff*. If this state was used in finding the optimal EAD label of another individual state (of the same or different bus), δ_{ead} of the later state would still be 1. Thus, eliminating the former state might increase the EAD labels but the delta values remain unaffected. Using LAD labels for reducing individual state space can be similarly justified.

Now consider phase II of elimination using the EAD labels. Since we may eliminate unmarked states, the delta values are likely to change in addition to the EAD labels. Depending on the new EAD labels, states which were originally unmarked (i.e. $\delta_{ead} = 0$) can get marked. For example, in the network from phase I shown in figure 4.6 on page 40, removing state (4,32) and the arc between (3,20) and (4,32) may preclude the possibility of reaching the destination within the *cutoff* $w.p. > 0$ from some or all of its ancestors. Thus we may iterate between the calculation of the EAD labels and the elimination phases until no states are excluded. Although we have not investigated the advantages of doing so in our computational experiments, we believe that this can further reduce the individual state space significantly. It is easy to see that the δ_{lad} values do not undergo any change after the second phase of elimination if LAD labels are used.

4.2.2 Elimination based on EOA labels

Let us now try to address the second question. If a traveler misses a bus, it might still be possible to catch the missed bus at a later stop using some faster service.

On this line of thought, we can employ the EOA labels to find the earliest possible arrival time at every node. If the latest time at which the bus arrives at a node is lesser than the EOA label, then the states of the bus at that node do not have any bearing on the optimal policy. This can be mathematically translated by defining a new indicator variable δ_{eoa} as follows:

$$\delta_{eoa}(s_b) = \begin{cases} 1 & \text{if } \tilde{t}(s_b) < \lambda_{\tilde{n}(s_b)} \\ 0 & \text{otherwise} \end{cases}$$

Consider the network shown in figure 4.4 on page 37 to illustrate the use of the δ_{eoa} indicator variable. Let the vector [15 25 35 50] represent the earliest we can reach nodes 1, 2, 3 and 4 respectively(note that the states at node 5 were discarded). The δ_{eoa} values for all states are set to 1 and hence we can disregard the bus while finding the optimal policy. Now consider another scenario in which the EOA labels are [15 25 30 42]. Clearly, we cannot arrive before the last possible states of the bus at nodes 1 and 2, but since the bus can be reached at node 3 $w.p. > 0$, it can be boarded at any other node along the remainder of its journey with positive probability. However, the states at node 1 and 2 cannot be eliminated as it leads to a loss of information (i.e., knowledge of the actual travel time on links (1,2) and (2,3) sheds more light on the state of the bus at subsequent nodes).

In such situations, we can completely ignore the bus if for each individual state, either δ_{eoa} or δ_{ead} is 1. Intuitively, this implies that we cannot catch the bus at some individual states, and in cases in which we can board the bus, the destination cannot be reached within the *cutoff*. A similar stronger but approximate scheme for elimination can be formulated by using the values of δ_{lad} instead of δ_{ead}.

4.2.3 Elimination based dominance

The third question posed earlier suggests the use of a dominance argument to eliminate the bus b_2. If FIFO order is preserved, bus b_2 can never arrive at a stop sooner than bus b_1. This idea can be traced back to first-order stochastic dominance as the probability of arriving at a node at a particular time using bus b_1 is always higher or equal to that of bus b_2. However, the fallacy of using such an argument stems from the fact that in minimizing the expected travel time, a traveler need not necessarily board a dominating bus. For instance, in the case of the two buses b_1 and b_2, the traveler could walk to another stop to board a faster service to the destination and return back to board bus b_2 in the event of missing the faster bus and still reduce the expected value. Thus elimination based on dominance is purely approximate.

4.3 Remarks on the elimination procedure

The methods to eliminate states descried earlier are not completely tight in the sense that they do not fully eliminate all the buses/states that do not affect the optimal policy. Consider the example shown in figure 4.8. Assume a traveler at node 1 at $t = 0$ is headed towards node 4. Let a bus start on each of the two routes at $t = 0$. It is easy to see that the value of λ_D is 20. Since the destination can be reached within the *cutoff* $w.p. > 0$ and $wp1$ from all individual states of the buses, no states are eliminated irrespective of the method used.

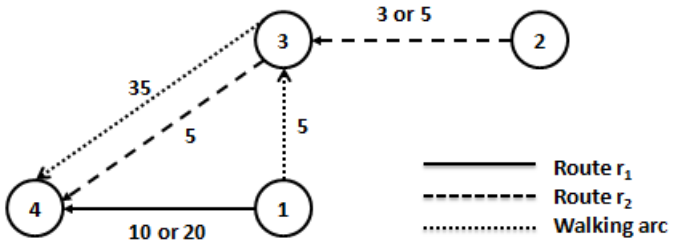

Figure 4.8: Drawbacks of the preprocessing techniques

However, in trying to board bus on route r_2 (which might be the optimal strategy for some particular pmfs) we risk reaching the destination beyond $t = 20$. Although we evade this issue by removing the walking arc between nodes 3 and 4 (explained later), we still have to deal with redundant states. Note that eliminating the states of the bus on route r_2 on the basis that it cannot be boarded $wp1$ might serve our purpose in this case. However, if a second bus on route r_1 starts at node 1 at $t = 10$ and takes 10 minutes to traverse arc (1,2), then for some particular pmfs, trying to catch the bus on route r_2 might be optimal.

4.4 Light Cones

The preprocessing methodology discussed in the previous section is heavily inspired from the idea of light cones in physics. A light cone is a flash of light from an event (E) that travels in spacetime. It comprises of two cones, a past and a future light cone. While, the past light cone represents events/points in spacetime from which a flash of light can be observed at E, the future light cone includes all points that can be reached by a light pulse from E.

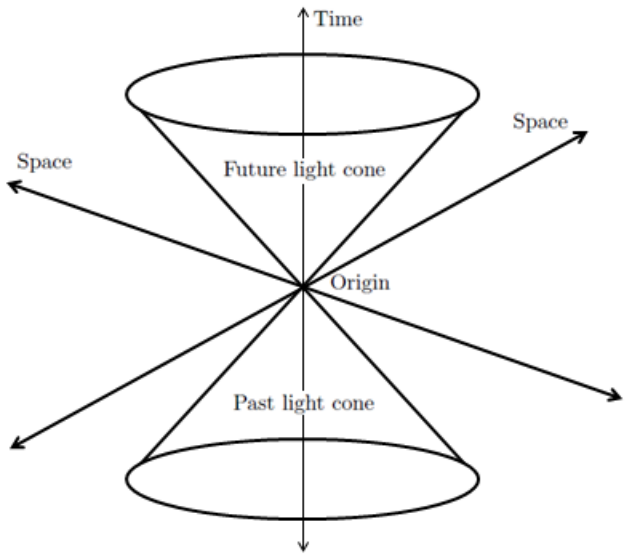

Figure 4.9: Light cones

Light cones serve as a perfect tool to understand causality, i.e., only the events which occur in the past light cone can possibly affect event E and event E can possibly influence only the events in the future light cone. Light cones in a two dimensional space can be represented as shown in the figure 4.9.

Finding buses/individual states that do not affect the optimal routing policy and eliminating them can be treated to be analogous to the concept of light cones. For illustrative purposes, we assume that the shape of a diagram based on how fast a traveler reaches other nodes in the network is a cone. Let us now draw a parallel between light cones and the elimination procedure using the example in figure 4.10. Suppose that the network comprises of a single bus and a traveler starting at the origin at $t = 0$ (having more buses in the network makes it impossible to pictorially represent higher dimensions). The cone at the origin can be obtained using the EOA labels.

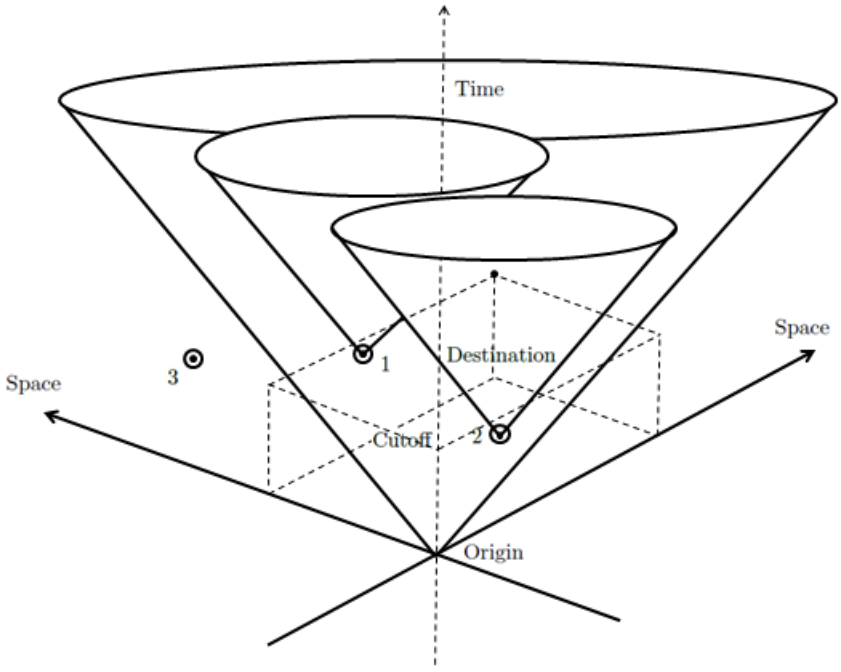

Figure 4.10: Light cones and indicator variables

Consider three individual states of a bus (1, 2 and 3) as shown in the figure. Let the cones at these states represent the earliest possible time taken to travel through the network. Individual state 3 lies outside the cone at the origin and hence cannot be reached. This corresponds to $\delta_{eoa}(3) = 1$. On the other hand, individual states 1 and 2 can be reached from the origin as they lie within the cone (i.e. $\delta_{eoa}(1) = \delta_{eoa}(2) = 0$). While the point represented by the destination and the *cutoff* lies inside the cone of state 2, it falls outside the cone of state 1. This is equivalent to the values of δ_{ead} being 1 and 0 for individual states 1 and 2 respectively.

Another abstract approach to visualize the same is to construct of a future cone from the origin at $t = 0$ and a past cone from the destination at $t =$*cutoff* as shown in figure 4.11. Buses whose individual states lie in the intersection of the two cones (see gray region) can be expected to influence the optimal strategy.

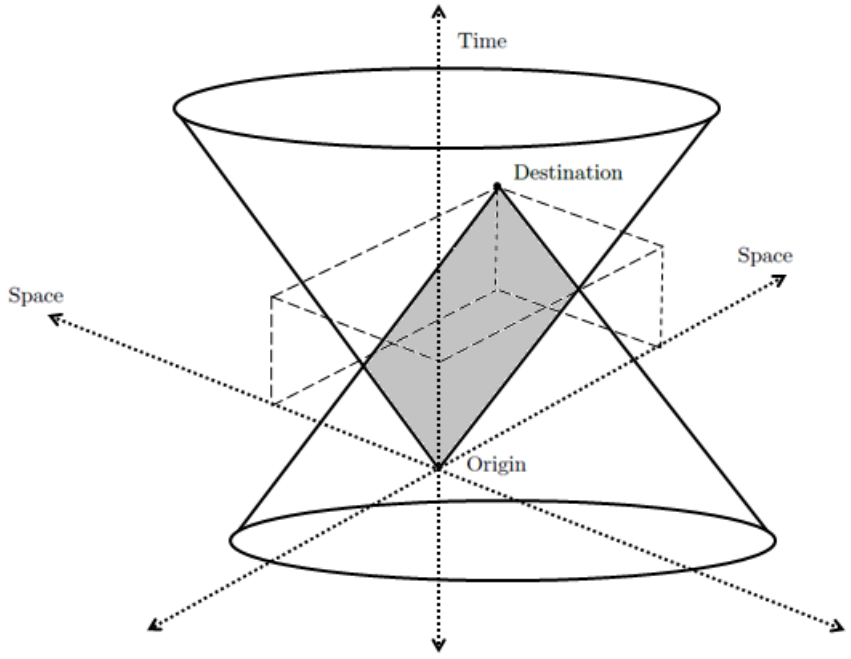

Figure 4.11: Relevant states of buses in the network

In a similar manner, we can relate the δ_{lad} variables and light cones. The cones obtained using the LAD labels can be imagined to be narrower compared to the ones resulting from the EAD labels, indicative of the fact that using LAD labels leads to the elimination of more states.

Chapter 5

Dynamic Programming framework for the ATR problem

In this chapter, we define the elements required to model the ATR problem as an MDP. The components of the dynamic program comprise of the state space, decision/action space, transition and value functions. Although an actual computation of the optimal strategy is not an objective of this thesis, we briefly discuss possible exact and approximation methods to solve this problem. Let us now examine these aspects in detail.

5.1 Components of Dynamic Program

5.1.1 State space

As defined earlier, the state space (S) is a subset of $\mathbb{S}$ where $\mathbb{S} = S_{b_1} \times S_{b_2} \times S_{b_3} ... \times S_{b_{|B|}} \times N \times T$. Note that the preprocessing methods reduce the sizes of the sets B and S_b and thus results in a smaller system state space. We may further redefine the set T to be $\{t_O, t_O + 1, ..., \lambda_D\}$. Let the states $(0, -1)$ and $(0, -2)$ be referenced as the *source* and *sink* respectively. Additionally, let t_{source_b} and t_{sink_b} denote the time at which the bus b enters the network and the earliest possible time at which it reaches the garage respectively. If a bus is already present in the network, the source is irrelevant and t_{source_b} is set to 0.

Consider the individual state network (see figure 5.1) obtained as a result of eliminating states using EAD labels. The network is identical to the one in figure 4.6 on page 40, but includes the time taken to travel between states. Let the cost of

an arc from state s_b, where $s_b \in S_b$ to state s'_b, where $s'_b \in \hat{\Gamma}(s_b)$(the set of successor/downstream states) be defined as follows.

$$\alpha(s_b, s'_b) = \begin{cases} 0 & \text{if } \hat{\Gamma}(s_b) = \{(0,-2)\} \\ \tilde{t}(s'_b) - \tilde{t}(s_b) & \text{if neither } s_b \text{ or} s'_b \text{ denotes the source or the sink} \\ \max\limits_{s''_b \in \hat{\Gamma}(s_b) \setminus \{s'_b\}} \alpha(s_b, s''_b) & \text{if } |\hat{\Gamma}(s_b)| \geq 2 \text{ and } s'_b = (0,-2) \\ t_{source_b} & \text{if } (0,-1) \in S_b \text{ and } s_b = (0,-1) \end{cases}$$

The first and the second cases in the above definition are trivial. Consider the third case in which a state is directly connected to the sink and has more than one outgoing arc. For example consider the arc between states (1,2) and $(0,-2)$ in figure 5.1. We know that if the travel time on arc between bus stops 1 and 2 is not 10, the destination cannot be reached within the *cutoff* and hence we assume the bus leaves the network. By convention, we set the cost of this arc to the maximum of the cost of all other outgoing arcs from state (1,2). The fourth case is again a convention we follow to simplify the computation of transition functions.

Associated with each arc is a parameter $\pi(s_b, s'_b)$ which is probability with which state s'_b can be reached from s_b. This may be calculated using the distribution of $\Omega_{a_{\beta_b}}$. However, if one of the states is the sink or source state then we write $\pi(s_b, s'_b)$ as follows:

$$\pi(s_b, s'_b) = \begin{cases} 1 & \text{if } \hat{\Gamma}(s_b) = \{(0,-2)\} \\ 1 & \text{if } (0,-1) \in S_b \text{ and } s_b = (0,-1) \\ 1 - \sum\limits_{s''_b \in \hat{\Gamma}(s_b) \setminus \{s'_b\}} \pi(s_b, s''_b) & \text{if } |\hat{\Gamma}(s_b)| \geq 2 \text{ and } s'_b = (0,-2) \end{cases}$$

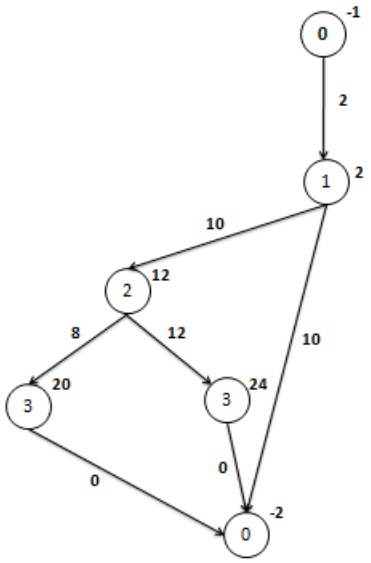

Figure 5.1: Constructing the state space

Notice that if a traveler is in the network at some node at say $t = 15$, the bus in the above figure cannot be at states (1,2), (3,20) and (3,24). It cannot be at the latter two states as they represent points in future. A bus at state (1,2) would have advanced to one of its successor states by $t = 12$. Hence the bus can only be present at either (2,12) or $(0, -2)$. More generally, for any time $t \in T$ the procedure in algorithm 4 may be used to mark the individual states at which the bus is likely to be present.

The algorithm first compares t with the time at which the bus enters the network. If $t < t_{source_b}$, the bus is still at the garage. Hence, we mark the source and stop. For example, in the network shown in figure 5.1, at $t = 1$, the bus can only be present at $(0, -1)$. As mentioned earlier, we could have multiple source states for each time step before the bus enters the network, but we find it unnecessary to do so as it increases the size of the state space.

Algorithm 4 *Marking individual states*

if $t < t_{source_b}$ **then** ▷ Condition I
  Mark source and stop
else
  for all states $s_b \in S_b \backslash \big\{\{(0,-1),(0,-2)\} \cup \{s_b : \hat{\Gamma}(s_b) \neq \{(0,-2)\}\big\}$ **do**
    if $t = \tilde{t}(s_b)$ **then** ▷ Condition II
      Mark state s_b
    else if $t > \tilde{t}(s_b)$ and $t < \tilde{t}(s_b) + \max_{s'_b \in \hat{\Gamma}(s_b)} (\alpha(s_b, s'_b))$ **then** ▷ Condition III
      Mark state s_b
    end if
  end for

  if $t \geq t_{sink_b}$ **then** ▷ Condition IV
    Mark sink
  end if
end if

Next, we scan all states except the source, sink and the states from which the bus immediately proceeds to the sink. If t coincides with the time at which the bus is present at a state we mark it using condition II. Condition III ensures that a state in future is never marked and the bus at a particular state is marked if it doesn't advance to one of its descendant states $wp1$. Finally, if t happens to be greater than or equal to t_{sink_b}, the bus is likely to be present in the garage and hence we mark the sink using condition IV.

Using the algorithm we first mark states for each bus. The set of system states at time t are then created by taking the cartesian product of the set of all marked states of each bus and the set N. Finally, we repeat this procedure for all $t \in T$ to obtain the entire the system state space S.

5.1.2 Decision space

We now move on to the next component of the dynamic program - the decision space. The decision space consists of the set of available actions at a particular state. Let us

denote the decision at a particular state by $x(s)$ and the set of all available actions by $X(s)$. At each state, recognize that a traveler can walk, wait or board a bus when it arrives. Thus, we define $X(s)$ as $X_w(s) \cup X_p(s) \cup X_r(s)$, where $X_w(s), X_p(s)$ and $X_r(s)$ comprise of the walking, waiting (loops of cost 1) and transit arcs available at state s. The set of available walking arcs, $X_w(s)$ may be defined as follows:

$$X_w(s) = \big\{(i,j) : (i,j) \in A_w,\, i = n \;\&\; t + w_{ij} \leq \lambda_D\big\}$$

The condition $i = n$ ensures that only walking arcs that emanate from node n are included. Additionally, to satisfy the risk aversion condition, we make sure that the arrival time at node j is less than the *cutoff*. An example of this has been discussed earlier using figure 4.8 on page 44 in which we exclude the walking arc between nodes 3 and 4 for all states at node 3. Using the shortest walking arcs between every pair of nodes is limiting in the sense that a traveler is expected to traverse the entire walking arc before making his/her next decision. In practice, one might choose an alternate strategy while walking between a pair of nodes based on the latest information of buses in the network. This feature can be easily modeled by considering a denser walking arc network. The set of waiting arcs can be similarly defined as shown below.

$$X_p(s) = \big\{(n,n) : t + 1 \leq \lambda_D\big\}$$

Now consider the transit options available at state s. Clearly, a transit arc can be traversed only if the individual states of a bus and that of a traveler match. Note that in choosing a transit service at a node, a traveler might not only choose the arc but also the bus which serves the arc. Hence we write $X_r(s)$ as $\bigcup_{b \in B} X_r^b(s)$, where $X_r^b(s)$ is the transit arcs available using bus b.

$$X_r^b(s) = \big\{(i,j) : (i,j) \in A_r,\, i = n = \tilde{n}(s_b),\, t = \tilde{t}(s_b),\, (0,-2) \notin \hat{\Gamma}(s_b),\, s_b \neq (0,-1)\big\}$$

In the above definition, conditions $n = \tilde{n}(s_b)$ and $t = \tilde{t}(s_b)$ imply that the traveler can catch bus b. Additionally, we also ensure that after boarding bus b, the traveler can get off at a node in the network $wp1$ by checking if the sink state isn't one of the successors of state s_b. For example, in the acyclic network of individual states shown in figure 5.1 on page 51, we may observe that arc (2,3) is the only transit arc that may be chosen if a traveler and the bus are at node 2 at $t = 12$.

Having defined the decision space, we now turn our attention to the cost of making a decision. While the cost of choosing to walk/wait is known deterministically, the actual travel time incurred by choosing a transit arc is random and is defined by the distributions $\Omega_{a_{\beta_b}}$. In general, let $\tilde{\xi}_{x(s)}$ be a random cost variable associated with a decision $x(s)$. Further, let $\xi_{x(s)}$ and $\Xi_{x(s)}$ denote a generic realization and the support of $\tilde{\xi}_{x(s)}$ respectively.

5.1.3 Transition functions

Given the decision made at a state, the transition functions for an MDP are used to describe the evolution of the system. More precisely, it gives the probability of ending up in a state s', assuming that a decision maker incurs a cost of $\xi_{x(s)}$ by choosing $x(s)$ at a state s. We denote the transition functions by $\mathbb{P}\big[s'|(s, x(s), \xi_{x(s)})\big]$, where $s, s' \in S, x(s) \in X(s)$ and $\xi_{x(s)} \in \Xi_{x(s)}$.

The fact that the uncertainty in travel time of the buses in the ATR problem is assumed to be independent of each other (see assumption 7) can be exploited in calculating the probabilities of future states of each bus separately, i.e. $\mathbb{P}\big[s'_b|(s_b, x(s), \xi_{x(s)})\big]$ where $s_b, s'_b \in S_b$. Further, owing to its repeated use it would be wise to calculate these probabilities and store it as a lookup table before solving the MDP.

We now briefly describe the process of finding the transition functions using backward induction on the network shown in figure 5.1 on page 51. In the following discussion,

successor states of a particular state s_b has been denoted with a hat and successors of successors of s_b are represented with a double hat. Assuming that the time elapsed from the instant a bus b arrives at a state s_b is denoted by t_e, let $\Pr\left[s'_b|(s_b, t_e, \hat{s}_b)\right]$, where $s'_b \in S_b$ and $\hat{s}_b \in \hat{\Gamma}(s_b)$ denote the probability with which the bus can be found in state s'_b given that $\pi\left[\left(s_b, \hat{s}'_b\right)|\left(s_b, t_e, \hat{s}_b\right)\right] = 0 \; \forall \; \hat{s}'_b \in \hat{\Gamma}(s_b) : \alpha(s_b, \hat{s}'_b) < \alpha(s_b, \hat{s}_b)$. In order to better understand the process of calculating these probabilities let us consider a few examples. Let the traveler be at some node in the network at $t = 15$, when the state of the bus is (2,12). Based on this information we can be certain that none of the downstream states of (2,12) are realized. Instead, if the traveler was at some node at say $t = 22$, we know that the bus will reach node 3 at $t = 24$ $wp1$. In other words, we may think of this as a scenario in which the state (3,20) is realized. Suppose the downstream states of an individual state are ordered in the increasing order of α (which is simply referred as arc costs), then $\pi\left[\left(s_b, \hat{s}'_b\right)|\left(s_b, t_e, \hat{s}_b\right)\right]$ denotes the probability of reaching one of the successor states of state s_b, given that all states with arc costs less than the arc cost of reaching a particular state $\hat{s}_b$ are realized. In the example under consideration, assuming that $\pi\big((2,12),(3,20)\big) = p$, we may write

$$\pi\left[\big((2,12),(3,20)\big)|\big((2,12),t_e,(3,20)\big)\right] = p$$

$$\pi\left[\big((2,12),(3,24)\big)|\big((2,12),t_e,(3,20)\big)\right] = 1-p$$

$$\pi\left[\big((2,12),(3,20)\big)|\big((2,12),t_e,(3,24)\big)\right] = 0$$

$$\pi\left[\big((2,12),(3,24)\big)|\big((2,12),t_e,(3,24)\big)\right] = 1$$

The first two equations correspond to the case where no downstream states are realized, while the next two equations corresponds to the situation in which one of the states i.e., (3,20) is realized. Clearly, for all t_e, $\Pr\left[s'_b|\big((0,-2),t_e,\hat{s}_b\big)\right] = 1$ if s'_b is the sink state and 0 otherwise. Proceeding backwards (examining states with the largest order first), for all $s_b \in S_b \backslash \{(0,-2)\}$ we can write a recursive equation as follows:

$$\Pr\left[s'_b|(s_b,t_e,\hat{s}_b)\right] = \sum_{\substack{\hat{s}'_b\in\hat{\Gamma}(s_b):\\ \alpha(s_b,\hat{s}'_b)\geq\alpha(s_b,\hat{s}_b)}} \frac{\pi(s_b,\hat{s}'_b)}{1-\sum_{\substack{\hat{s}''_b\in\hat{\Gamma}(s_b):\\ \alpha(s_b,\hat{s}''_b)<\alpha(s_b,\hat{s}_b)}}\pi(s_b,\hat{s}''_b)} \Pr\left[s'_b|(\hat{s}'_b,(t_e-\alpha(s_b,\hat{s}'_b)),\hat{\hat{s}}_b)\right]$$

where $\hat{\hat{s}}_b \in \hat{\Gamma}(\hat{s}'_b)$, is a successor state of $\hat{s}'_b$, such that $\alpha(\hat{s}'_b,\hat{\hat{s}}_b)$ has the lowest cost among all arcs from state $\hat{s}'_b$. Also, we follow the convention that if $\big(t_e-\alpha(s_b,s'_b)\big)<0$,

$$\Pr\left[s'_b|\big(\hat{s}'_b,(t_e-\alpha(s_b,\hat{s}'_b)),\hat{\hat{s}}_b\big)\right] = \begin{cases} 1 & \text{if } s'_b = s_b \\ 0 & otherwise \end{cases}$$

Let us now try to parse the above expressions. Assuming that the bus is allowed to travel for t_e minutes from a state s_b, we consider only downstream states ($\hat{s}'_b$) which aren't realized. The posterior probabilities of reaching these states are then computed using the first expression in the summation. If the cost of an arc that connects s_b to a state that hasn't been realized is higher than t_e, we assume that the bus travels for $t_e-\alpha(s_b,\hat{s}'_b)$ minutes from state $\hat{s}'_b$. Using the state $\hat{\hat{s}}_b$, we ensure that the none of the successor states of state $\hat{s}'_b$ are realized. The probabilities of reaching other states are then obtained using the previously calculated values of $\Pr\left[s'_b|(\hat{s}'_b,t_e-\alpha(s_b,\hat{s}'_b),\hat{\hat{s}}_b)\right]$. The case in which the time elapsed is less than the cost of arc from state s_b to $\hat{s}'_b$ is taken care by the assumed convention.

Using precomputed values of the above probabilities (for all t_e bounded by the largest possible travel time on any arc), for all buses that are not involved in the decision $x(s)$, assuming $\tilde{t}(s_b)$ is set to 0 if $s_b=(0,-1)$ or $(0,-2)$, the values of $\mathbb{P}\big[s'_b|(s_b,x(s),\xi_{x(s)})\big]$, where $s_b,s'_b \in S_b$ can be looked up from $\Pr\left[s'_b|\big(s_b,(t-\tilde{t}(s_b)+\xi_{x(s)}),\hat{s}_b\big)\right]$ where $\hat{s}_b = \underset{\substack{\hat{s}'_b\in\hat{\Gamma}(s_b):\\ \alpha(s_b,\hat{s}'_b)+\tilde{t}(s_b)\geq t}}{\operatorname{argmin}} \big[\alpha(s_b,\hat{s}'_b)+\tilde{t}(s_b)-t\big]$.

5.1.4 Value functions

The value function of a state (denoted by $V(s)$) is the expected time taken to reach the destination from that particular state. The value function for all states in which n is the destination node is set to 0 and the value of all other states is set to ∞.

5.2 Solving the Dynamic Program

Using the definitions and notation described so far, the Bellman's equation of optimality can be expressed as follows:

$$V(s) = \max_{x(s) \in X(s)} \left[\mathbb{E}_{\Xi_{x(s)}} \left[\tilde{\xi}_{x(s)} + \sum_{s' \in S} \mathbb{P}\left[s' | (s, x(s), \tilde{\xi}_{x(s)}) \right] V(s') \right] \right]$$

In order to solve for the optimal labels, one can use the value iteration method (see Powell [23]), which may be described as follows:

Algorithm 5 *Pseudocode for Value Iteration*

Step 0: Initialize value functions
$V(s) = 0 \ \forall \ s : n = D$
$V(s) = \infty \ \forall \ s : n \neq D$
$\tau = \lambda_D$

Step 1:
while $\tau \geq t_O$ **do**
  for all $s \in S : t = \tau$ **do**

$$V(s) = \max_{x(s) \in X(s)} \left[\mathbb{E}_{\Xi_{x(s)}} \left[\tilde{\xi}_{x(s)} + \sum_{s' \in S} \mathbb{P}\left[s' | (s, x(s), \tilde{\xi}_{x(s)}) \right] V(s') \right] \right]$$

  end for
  $\tau = \tau - 1$
end while

The size of the state space might still be large after the preprocessing stage making it unattractive for using the value iteration method to solve the ATR problem. In such cases it could be worthwhile to explore approximate dynamic programming methods

in which value functions may be approximated using sampling methods or aggregation of arcs and pmfs.

Chapter 6

Results and Conclusions

In this chapter, the results of the numerical experiments are presented along with conclusions and scope for future work. The computational analysis comprised of a study of the state space reduction on a small instance of the Austin transit network.

6.1 Computational experiments

6.1.1 Network Description

Eight routes on the Austin transit network were chosen, and information available on the Capital Metropolitan Transportation Authority's (Capital Metro) website was used for this study. The following table shows the summary of the input data.

Table 6.1: Input Data

Network Characteristics	
No. of Routes	8
No. of Buses	48
No. of Nodes	78
Routes	3, 5, 7, 10 (NB and SB)
Period of Interest	6:00 AM - 12:00 PM

The figure 6.1 shows the routes and time points and figure 6.2 shows the schedules at various time points along routes 10 NB and 10 SB. All trips that begin between 6:00 AM and 12:00 PM were included in the model. A total of 78 stops were considered, and the shortest walking arc time between each pair of nodes was obtained using the Google Distance Matrix API. The implementation was carried out in C++ on a Linux machine with a 6 core Intel Xeon processor (3.33 GHz).

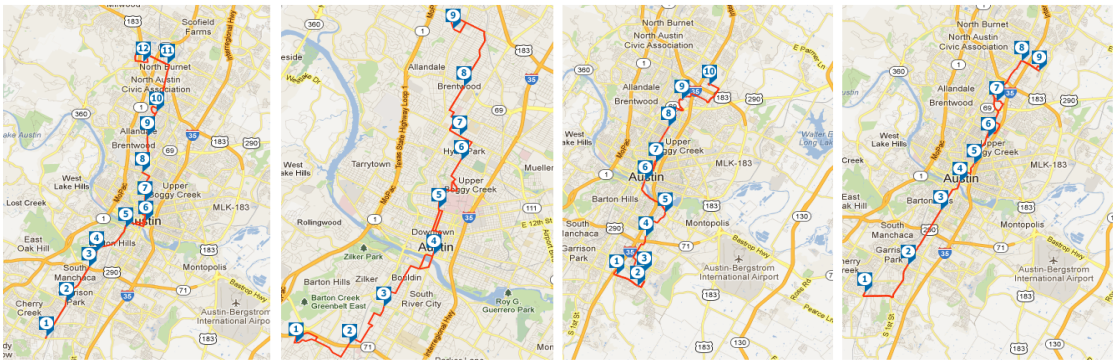

Figure 6.1: Routes (Source : http://www.capmetro.org/)

Trips on routes were manually assigned to buses (as this information was not available), using which itineraries for each bus were constructed. In some cases, buses were assigned to trips across several routes (for example buses on route 10 also serve route 3, see fig 6.2). A current state vector and pmfs on transit arcs were randomly generated (the size of the support of pmfs on each transit link was restricted to two) and were used to construct the individual states of each bus in the network.

10 WEEKDAYS/NORTHBOUND

Manchaca at Slaughter ❶	S. 1st at Wm. Cannon ❷	S. 1st at Oltorf ❸	Congress at 2nd ❹	Red River at MLK ❺	Hancock Center ❻	Highland Mall ❼	Georgian at Rundberg ❽	Aberdeen at Rundberg ❾	To Route/Garage	Special Note
5:09	5:20	5:34	5:42	5:54	6:02	6:17	6:29	6:37		DL
5:39	5:50	6:04	6:12	6:24	6:32	6:47	6:59	7:07		DL
5:59	6:14	6:32	6:42	6:55	7:05	7:16	7:28	7:37		
6:21	6:36	6:54	7:04	7:17	7:27	7:38	7:50	7:59		
6:43	6:58	7:16	7:26	7:39	7:49	8:00	8:12	8:21		
7:05	7:20	7:38	7:48	8:01	8:11	8:22	8:34	8:43		
7:27	7:42	8:00	8:10	8:23	8:33	8:44	8:56	9:05		
7:49	8:04	8:22	8:32	8:45	8:55	9:06	9:18	9:27		
8:11	8:26	8:44	8:54	9:07	9:17	9:28	9:40	9:49		
8:33	8:48	9:06	9:16	9:29	9:39	9:50	10:02	10:11	G	
8:59	9:12	9:29	9:38	9:53	10:03	10:15	10:28	10:36		
9:25	9:38	9:55	10:04	10:19	10:29	10:41	10:54	11:02		

10 WEEKDAYS/SOUTHBOUND

Aberdeen at Rundberg ❾	Georgian at Rundberg ❽	Highland Mall ❼	Hancock Center ❻	Red River at 18th ❺	11th at San Jacinto ❹	S. 1st at Oltorf ❸	S. 1st at Wm. Cannon ❷	Manchaca at Slaughter ❶	To Route/Garage
—	—	4:29	4:40	4:49	4:53	5:06	5:17	5:28	3
4:40	4:46	4:59	5:10	5:19	5:23	5:36	5:47	5:58	3
5:10	5:16	5:29	5:40	5:49	5:53	6:06	6:17	6:28	3
5:40	5:46	5:59	6:10	6:19	6:23	6:36	6:47	6:58	3
6:01	6:08	6:22	6:35	6:46	6:51	7:07	7:22	7:33	3
6:23	6:30	6:44	6:57	7:08	7:13	7:29	7:44	7:55	3
6:45	6:52	7:06	7:19	7:30	7:35	7:51	8:06	8:17	3
7:07	7:14	7:28	7:41	7:52	7:57	8:13	8:28	8:39	3
7:29	7:36	7:50	8:03	8:14	8:19	8:35	8:50	9:01	3
7:51	7:58	8:12	8:25	8:36	8:41	8:57	9:12	9:23	3
8:13	8:20	8:34	8:47	8:58	9:03	9:19	9:34	9:45	3
8:35	8:42	8:56	9:09	9:20	9:25	9:41	9:56	10:07	3

Figure 6.2: Schedules on Route 10 (Source : http://www.capmetro.org/)

6.1.2 Numerical results

Table 6.2 shows the number of individual states of buses in the network at the end of each step in the preprocessing procedure for an instance in which the origin and

departure time of the traveler were randomly chosen.

Table 6.2: Results of individual state space elimination

Bus ID	*Before elimination*	*Risk aversion*	*Phase I (EAD)*	*Phase II (EAD)*	*Phase I (LAD)*	*Phase II (LAD)*
1	1178	19	11	8	8	8
2	5308	26	15	10	15	10
3	3625	19	-	-	-	-
4	3750	4	-	-	-	-
6	5200	22	18	14	18	14
7	5299	15	-	-	-	-
8	3476	5	-	-	-	-
9	3887	31	8	8	8	8
10	4978	26	11	5	11	5
11	2724	18	-	-	-	-
12	2798	5	-	-	-	-
14	4769	12	7	6	7	6
15	2779	8	-	-	-	-
16	2888	4	-	-	-	-
17	220	2	-	-	-	-
18	3649	2	-	-	-	-
20	2628	30	4	4	4	4
21	3752	15	10	8	10	8
22	3983	5	-	-	-	-
23	3115	19	8	8	8	8
24	3844	19	-	-	-	-
25	4218	3	-	-	-	-
26	672	15	10	9	5	-
27	4584	11	-	-	-	-
28	2979	12	-	-	-	-
29	2908	5	-	-	-	-
30	2882	4	-	-	-	-
33	4199	15	8	8	8	8
34	2547	17	16	16	16	16
35	1093	14	-	-	-	-
36	2843	7	-	-	-	-
37	2930	3	-	-	-	-
38	2885	2	-	-	-	-
Product	4.26E+148	9.99E+31	7.49E+11	7.93E+10	2.72E+11	8.81E+09

It was found that the destination could be reached within 48 minutes $wp1$. Buses which were completely eliminated based on the risk aversion criteria are not shown in the table. Although, the results of the phase I and II elimination procedure appears promising when compared to the total number of initial individual states, it would be only be fair to compare them with the size of the individual state space obtained after the risk aversion constraints are imposed. The plot in figure 6.3 shows the logarithm of the cardinality of the cartesian product of the individual state space (which is proportional to the size of the actual state space) for various preprocessing steps. While the magnitude of reduction in state space is significant using the phase I, further reduction due to phase II appears to be marginal. As mentioned earlier it might be possible to further reduce the state space by recursive application of the Phase I and Phase II methods. Note that the actual state space is constructed as discussed in section 5.1.1 on page 49 and is much smaller than the values used for comparison.

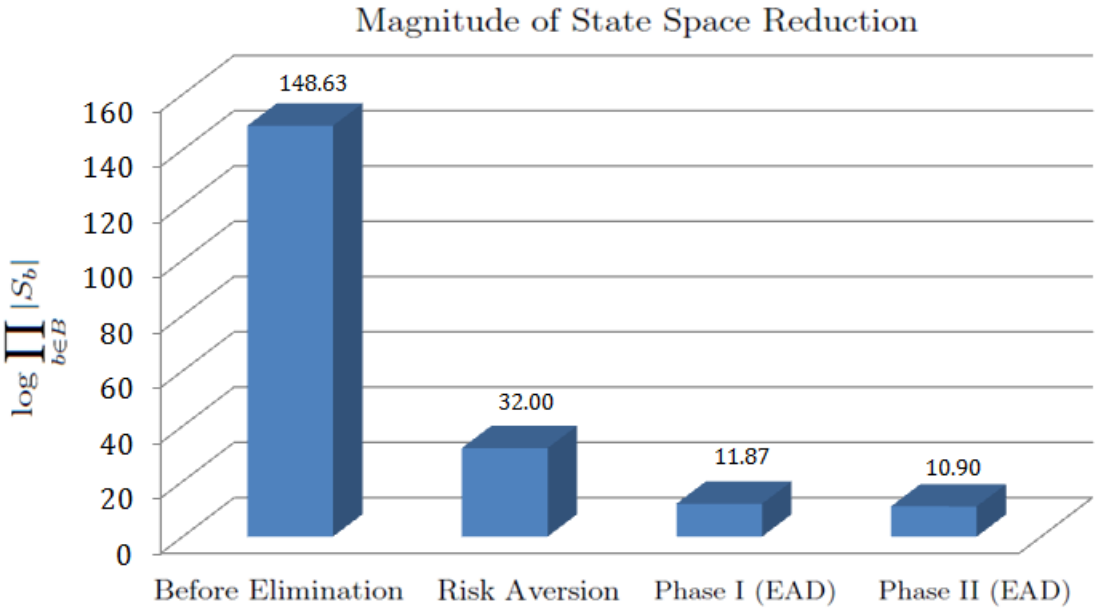

Figure 6.3: A Plot of the individual state space reduction

We further tested the reduction in the individual state space using the LAD labels and found that this method aids in eliminating more states as expected. A comparison of the EAD and LAD elimination procedure is shown in the following graph.

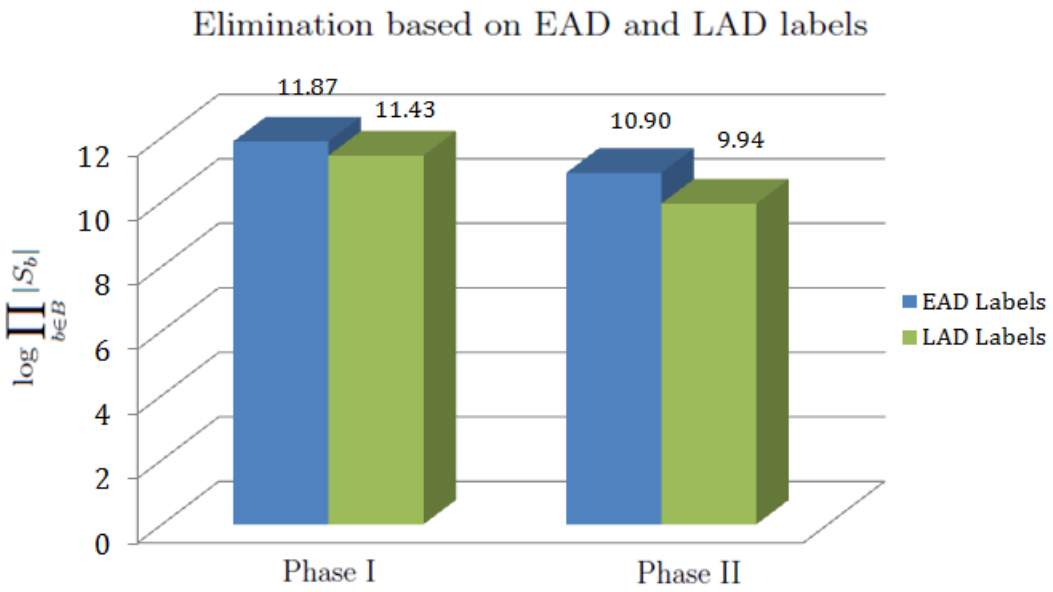

Figure 6.4: Comparison of individual state space reduction using EAD and LAD labels

The experiments were repeated for another instance in which the destination was guaranteed to be reached within 216 minutes (two extreme ends of the city were chosen for this purpose). Since the cutoff was higher, relatively lesser states were eliminated using the risk aversion condition and the elimination methods, but more states were found to be eliminated using the LAD labels when compared to the elimination based on EAD labels. We did not observe any reduction in the state space using arguments based on the EOA labels. A summary of the results for this scenario is shown in the following plot.

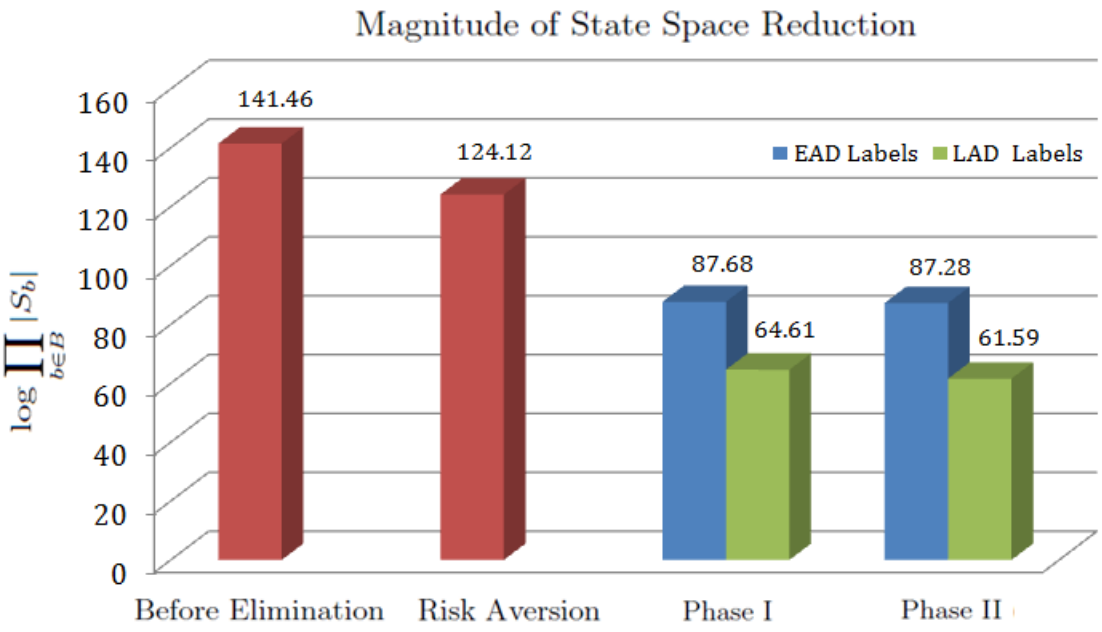

Figure 6.5: Plot of individual state space reduction for another OD pair

The results in general are influenced by several factors such as the OD pair, density of buses in the network and the probability distributions, which in turn affect the *cutoff* and the EAD and LAD labels. The numerical results presented here serve as a rough reflection of the abilities of these approaches and it is difficult to expect similar results if these methods are replicated on a different network. For instance, one might assume that having a denser network of buses makes it easier to reach the destination from individual states within the *cutoff* and hence, lesser number of states may be eliminated. But the LOA algorithm in a dense transit network might yield a smaller *cutoff* in the first place. The computation time for both examples was observed to be well under a minute.

6.2 Conclusions

In this thesis, an adaptive transit routing problem in which a traveler seeks a strategy that minimizes the expected cost of travel and ensures that the destination is reached

within a certain threshold was addressed. A strategy is a set of actions that are conditional on the state of all buses and the traveler in the network. The state of a bus is defined by the most recently visited stop and the time at which the bus visited it. Using discrete probability distributions of link travel times a framework for formulating the problem as an MDP was developed.

In practice, a major challenge in using this approach involves working with a large system state space. This has been widely referred to as the *curse of dimensionality* in the field of dynamic programming. In this thesis, attempts were made to reduce the system state space by independent reduction of the individual state space of each bus. This process was facilitated by some algorithms and assumptions on the behavioral attitudes of travelers. Further, some numerical experiments were performed to gauge the potential of these methods.

The ATR problem addresses the effects of congestion and its impacts on route choice and transfers using adaptive strategies in a stochastic time dependent transit network. Some of the major contributions of this thesis include ability of using a vast amount of real time information that can potentially result in lower expected travel times compared to an a priori strategy. In addition, state space reduction methods that may be applied independently on each bus in the network were developed along the lines of causality and the concept of light cones. Also, the bus based approach and assumptions made let us model the strategies and the slack in schedules between trips in a much broader and realistic manner.

However, the proposed model is limited by the assumptions made, some of which are restrictive in nature. For instance, the travel time distributions on links are assumed to be independent of each other. While this may be true for delays caused due to passenger boarding and alighting, buses on routes that share links are likely to

be affected in a similar manner in the presence of congestion. Another assumption that limits the scope of the ATR problem has to do with a traveler's reluctance in transferring multiple times before reaching the destination. However, enforcing constraints on the number of transfers in the present model is extremely difficult. Further, the effects of fares and congestion are ignored in the decision making process.

6.3 Scope for future study

The study of the ATR problem presents some useful pointers for future research on this topic. While the preprocessing methods look attractive, an exact estimation of the optimal strategy may be hindered by the size of the state space depending on factors such as the number of buses, the origin and the destination, the pmfs on links, etc. In such cases, one would have to resort to approximation techniques, in which case it is necessary to investigate the quality of solutions obtained using these methods.

Another approach to address the ATR problem is to treat it as a Knapsack Problem in which buses may be regarded as items. Each bus may not only contribute to a decrease in the expected value, but the combination of buses chosen might further bring us closer to the optimal solution. Cardinality constrained versions of this problem can restrict the size of the state space to an extent that the ATR problem may be approximately solved using exact methods. Although, greedy algorithms that exploit the sub modular nature of such problems are known to exist, quantifying the benefits of including a bus or a subset of buses can be quite challenging.

It is also of prime interest to determine the travel time savings in using these methods in comparison with an a priori strategy or other simpler trip planning approaches. As noted earlier in the case of the benefits of the state space reduction, the savings in travel time are also dependent on the OD pair, pmfs and the density of the transit

network, etc. For instance, in the example used to illustrate the adaptive nature of the ATR problem in chapter 1(see figure 1.1 on page 2), the travel time savings is 3 min, which is a 40 percent decrease over the expected travel time of the optimal a priori strategy. Instead, if the travel time on the link (4,5) is either 1 or 2, the optimal cost of the a priori strategy and the adaptive solution are the same. Hence, it is very difficult to judge the benefits of using this model which calls for extensive testing on transit networks with accurate probability distributions and information of bus itineraries.

The elimination process offers scope for parallelization as the procedure developed may be independently applied to each bus. Also, there exists more room for reduction of the state space as noted in discussions on the limitations of the preprocessing methods in chapter 4. The ATR problem may further be extended to incorporate correlations in the link travel time distributions, which offers a more realistic representation of the problem. While this complicates the transition functions of the dynamic program, it may result in higher the EAD and EOA labels, and aid in the process of elimination of states to a greater extent. Although the ATR problem remains to be explored in greater detail, this study develops a sound theoretical framework and novel state space reduction techniques that contribute significantly to the study of adaptive routing in transit networks.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

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

[4] C. Chriqui and P. Robillard. Common bus lines. *Transportation Science*, 9(2):115, 1975.

[5] J. de Cea and E. Fernndez. Transit assignment for congested public transport systems: An equilibrium model. *Transportation Science*, 27(2):133–147, 1993.

[6] N. Deo and C.-Y. Pang. Shortest-path algorithms: Taxonomy and annotation. *Networks*, 14(2):275–323, 1984.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[8] S. E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395 – 412, 1969.

[9] L. R. Ford. Network flow theory. Technical Report P-923, Rand Corporation, Santa Monica, CA, 1956.

[10] G. Gentile, S. Nguyen, and S. Pallottino. Route choice on transit networks with online information at stops. *Transportation Science*, 39(3):289–297, 2005.

[11] R. W. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182 – 188, 1986.

[12] Y. Hamdouch and S. Lawphongpanich. Schedule-based transit assignment model with travel strategies and capacity constraints. *Transportation Research Part B: Methodological*, 42(78):663 – 684, 2008.

[13] M. D. Hickman. *Assessing the impact of real-time information on transit passenger behavior*. PhD thesis, Massachusetts Institute of Technology, 1994.

[14] M. D. Hickman. Robust passenger itinerary planning using transit avl data. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 840 – 845, 2002.

[15] M. D. Hickman and D. H. Bernstein. Transit service and path choice models in stochastic and time-dependent networks. *Transportation Science*, 31(2):129–146, 1997.

[16] M. D. Hickman and N. H. Wilson. Passenger travel time and path choice implications of real-time transit information. *Transportation Research Part C: Emerging Technologies*, 3(4):211 – 226, 1995.

[17] R. Huang and Z.-R. Peng. Schedule-based path-finding algorithms for transit trip-planning systems. *Transportation Research Record*, 1783:142– 148, 2002.

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

[19] 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.

[20] S. Nguyen and S. Pallottino. Equilibrium traffic assignment for large scale transit networks. *European Journal of Operational Research*, 37(2):176–186, 1988.

[21] S. Nguyen and S. Pallottino. Hyperpaths and shortest hyperpaths Combinatorial Optimization. In *Combinatorial Optimization*, volume 1403 of *Lecture Notes in Mathematics*, chapter 10, pages 258–271. Springer Berlin / Heidelberg, 1989.

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

[23] W. B. Powell. *Approximate Dynamic Programming : Solving the Curses of Dimensionality*. John Wiley & Sons, Hoboken, New Jersey, 2007.

[24] D. Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123(2):315–324, 2000.

[25] J. S. Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.

[26] H. Spiess and M. Florian. Optimal strategies: A new assignment model for transit networks. *Transportation Research Part B: Methodological*, 23(2):83–102, April 1989.

[27] M. C. Tan, C. O. Tong, S. C. Wong, and J. M. Xu. An algorithm for finding reasonable paths in transit networks. *Journal of Advanced Transportation*, 41(3):285–305, 2007.

[28] C. O. Tong and A. J. Richardson. A computer model for finding the time-dependent minimum path in a transit system with fixed schedules. *Journal of Advanced Transportation*, 18(2):145–161, 1984.

[29] S. T. Waller and A. K. Ziliaskopoulos. On the online shortest path problem with limited arc cost dependencies. *Networks*, 40(4):216–227, 2002.

[30] J. H. Wu, M. Florian, and P. Marcotte. Transit equilibrium assignment: A model and solution algorithms. *Transportation Science*, 28(3):193–203, 1994.

[31] W. Xu, S. He, R. Song, and S. S. Chaudhry. Finding the k shortest paths in a schedule-based transit network. *Computers and Operations Research*, 39(8):1812 – 1826, 2012.

[32] A. K. Ziliaskopoulos and H. S. Mahmassani. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record*, 1408:94–100, 1993.