Adaptive Transit Routing in Stochastic Time-Dependent Networks

Tarun Rambha[1], Stephen D. Boyles[1] and S. Travis Waller[2]

[1]Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin

[2]School of Civil and Environmental Engineering, University of New South Wales

Abstract

We define an adaptive routing problem in a stochastic time-dependent transit network in which transit arc travel times are discrete random variables with known probability distributions and formulate it as a finite horizon Markov decision process. Routing strategies are conditioned on the arrival time of the traveler at intermediate nodes, and real time information on arrival times of buses at stops along their routes. The objective is to find a strategy that minimizes the expected travel time, subject to a constraint that guarantees that the destination is reached within a certain threshold. While this framework proves to be advantageous over *a priori* routing, it inherits *the curse of dimensionality* and state space reduction through preprocessing is achieved by solving variants of the time-dependent shortest path problem. Numerical results on a network representing a part of the Austin transit system indicate promising reduction in the state space size and improved tractability of the dynamic program.

Keywords: transit routing; stochastic shortest paths; curse of dimensionality; state space reduction; Markov decision process

1 Introduction

Transit networks are often subject to uncertainty in arc travel times. Variability in the time taken to traverse an arc results from several factors such as congested road conditions, traffic signals, inclement weather and maintenance disruptions (particularly in rail networks). These factors play an important role in trips involving transfers and in cities with alternate transit options between an origin-destination (OD) pair. However, most transit routing applications seldom take uncertainty into account while providing routing policies. These applications prescribe what we call *a priori* strategies, which inform travelers where to board, get down and transfer based on frequencies or schedules published by transit agencies. *A priori* strategies in stochastic networks are generally sub-optimal, and it is possible to construct adaptive strategies with lower expected travel time by making use of information related to the location and travel times of buses in a network. Recent advances in intelligent transportation systems (ITS) allow us to gather such real-time data (such data is already publicly available in several cities around the world), which may also be used in characterizing the distributions of uncertainty in the network. .

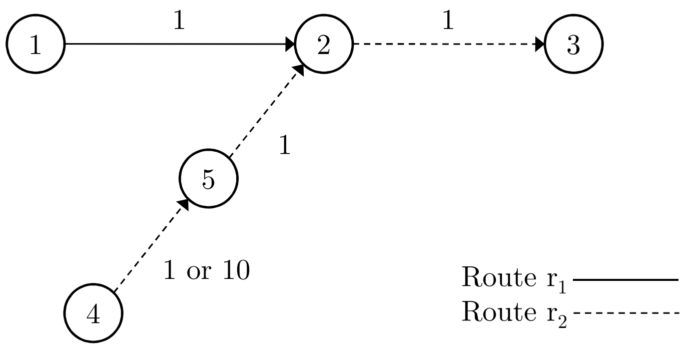


Figure 1: Illustration of adaptive routing in transit networks

Consider an example (see Figure 1) to illustrate the sub-optimality of *a priori* strategies. Let at $t = 0$, buses b_1 and b_2 start at nodes 1 and 4 on routes r_1 and r_2 respectively. The time taken by the buses to traverse the transit arcs is indicated in the figure. Assume that the travel time on the transit arc $(4, 5)$ is either 1 or 10 with equal probability. Also, assume that the walking travel time along the arcs shown in the network is 5. Suppose we wish to travel from node 1 to node 3. If we board bus b_1 and reach node

2, we could either walk to node 3 or wait for bus b_2. The expected travel time for these two strategies is 6 and 7.5 respectively. Therefore, the optimal *a priori* strategy is to board bus b_1 and walk to the destination. However, if bus b_2 reaches node 5 at $t = 1$, it is guaranteed to arrive at node 2 at $t = 2$ and hence waiting is optimal. But if, at $t = 1$, we received information that bus b_2 failed to reach node 5, walking to node 3 is optimal. Such an adaptive strategy has an expected cost (throughout this paper, the terms *travel time* and *cost* are used interchangeably) of $3(0.5) + 6(0.5) = 4.5$, which is lower than the optimal *a priori* solution.

Adaptive shortest path problems in stochastic networks have been widely studied in the literature. In his seminal paper on this subject, Hall (1986) noted that the least expected time path cannot be found using standard shortest path algorithms as it is not a simple path, but a strategy or a hyperpath in which arcs are chosen based on the arrival time at intermediate nodes. Miller-Hooks and Mahmassani (2000), and Miller-Hooks (2001) developed efficient labeling algorithms to solve the problem of finding the adaptive least expected path in stochastic time-dependent networks in which the arc travel time distributions vary with time. Pretolani (2000) solved a similar problem using Nguyen and Pallottino (1989)'s shortest hyperpath algorithms. Polychronopoulos and Tsitsiklis (1996) used dynamic programming to formulate stochastic shortest path problems with recourse in which arc costs are random and the uncertainty is revealed as the network is traversed. Waller and Ziliaskopoulos (2002), and Provan (2003) extended the problem of finding adaptive strategies to networks with arc cost dependencies under a reset assumption according to which the cost of an arc is realized every time its tail node is reached even if it was previously visited. Some of the other extensions of stochastic shortest path problems include K-shortest paths (Nielsen et al. (2004) and Nielsen et al. (2014)), bi-criterion shortest paths (Miller-Hooks and Mahmassani (1998), and Nielsen et al. (2003)), and optimal routing in the presence of correlated arc travel times (Huang and Gao (2012)). However, finding adaptive paths in transit networks is relatively difficult due to the possibility of waiting and the issue of *common bus lines* as noted by Chriqui and Robillard (1975). A traveler in a transit network is often faced with the option of boarding multiple buses on different lines to traverse an arc or a section of a route.

Adaptive route choice in transit networks has been primarily studied as a sub-problem in frequency based transit assignment. In these models, the headway between bus arrivals is assumed to be random with some known distribution (typically exponential). In the presence of common bus lines, travelers choose a subset of available lines, called the *attractive set*, such that the expected travel time to the destination is minimized. A strategy is defined using the line chosen at each stop (or a probability distribution over the attractive set) and the alighting point corresponding to the chosen line. While these models generally focus on estimating the expected waiting time, randomness in the in-vehicle travel time is ignored and transfers are not explicitly modeled. Spiess and Florian (1989) considered a transit network in which waiting times at nodes depend on the combined frequency of lines, and the strategy of a traveler is to board the first bus serving a line belonging to the attractive set. Nguyen and Pallottino (1988), and Nguyen et al. (1998) developed a graph theoretical framework in which travelers are assumed to choose hyperpaths, which define a strategy and provide the probability of boarding a line belonging to the attractive set. The cost of a hyperpath includes transit arc costs and waiting travel times weighted by appropriate probabilities. de Cea and Fernández (1993), and Wu et al. (1994) examined strategies in a broader class of transit assignment problems which incorporate the effects of congestion.

Route choice in stochastic time-dependent transit networks in the presence of online information was studied by Hickman (1994), Hickman and Wilson (1995), and Hickman and Bernstein (1997) using a dynamic path choice model in which a traveler at the origin, based on the information gained while waiting, decides to board a bus or wait for a bus that arrives later. 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, but also on the information on buses serving the node received until that node is reached. However, their proposed extension is somewhat limited as it only considers the information on buses that arrive at a particular node at which a traveler boards or transfers. Gentile et al. (2005) computed optimal routing strategies and equilibrium transit flows assuming that passengers have access to the estimated waiting time and the expected time to reach the destination for each line in the network. A more detailed summary of literature on adaptive routing in transit systems can be found in Rambha (2012).

In this paper, we develop an adaptive route choice model in schedule-based transit networks which are subject to uncertainty in arc travel times. We propose a less restrictive definition of a strategy by making use of information on all the buses in the network. A treatment of the adaptive transit routing (ATR) problem as a finite horizon Markov decision process (MDP) is facilitated by the definitions of the state of the system and the discretization mechanism. Strategies are described using the notion of system states (which are defined by the spatial and temporal locations of buses and the traveler in the network) and travelers are assumed to choose least expected cost policies which guarantee that the destination is reached within a prespecified time. However, this framework leads to an unwieldy state space, but not all system states are likely to influence the optimal strategy. Therefore, a major goal of this paper is to develop causality based preprocessing methods that help reduce the state space to improve tractability. For example, in a problem instance discussed later, the size of the state space is reduced roughly from about 10^{31} to 10^{10}. These preprocessing methods make use of results from proposed variants of the time-dependent shortest path problem. Labeling algorithms for solving these variants are also discussed. The performance of the models developed in this study makes a case for using bus-based approaches for transit routing as opposed to conventional line-based methods.

The rest of this paper is organized as follows: In Section 2, we introduce the problem and present the notation used. Section 3 contains a description of the preprocessing methods used to reduce the state space. In Section 4, we develop a framework for solving the ATR problem as an MDP. Section 5 contains the computational results of an implementation of the state space reduction methods and the dynamic program on a portion of the Austin, Texas transit network. Finally, in Section 6, we summarize the findings and limitations of this study, and discuss possible directions for future research.

2 Problem Description

2.1 Notation

Let B and R represent the set of buses and routes (in this paper, the terms *routes* and *lines* are assumed to have the same meaning) 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. Let $G = (N, A)$ be a directed network, where N represents the set of nodes/bus stops and A denotes the set of arcs. If a node happens to be the destination of one route and the origin of another and if a single bus serves the two routes in succession, the node is replicated. We define A as $A_w \cup A_r \cup A_d$, where A_w represents the set of walking arcs between every pair of nodes in N (includes self-loops of cost 1 to model waiting), A_r consists of transit arcs between bus stops along routes in the network, and A_d represents the set of dummy arcs used to model the slack in schedules when buses switch routes (discussed in detail in Section 2.3). Slack here refers to the layover time between the trips made by a bus. The time period of interest T is divided into unit intervals $\{0, 1, 2, \ldots, t, \ldots, T_{max}\}$ each of which denotes the time elapsed from the start of the first trip of first bus. Let t_O be the time at which a traveler departs from the origin node O. We do not set t_O to 0 as we need to keep track of buses that are already in the network when the traveler departs from the origin.

Let β_b describe a trip (which is assumed to contain information related to stops and scheduled arrival times) along a route served by bus $b \in B$. We denote the set of *individual states of a bus* b by S_b and define its elements (represented by s_b) using the ordered pair (n_b, t_b), where $n_b \in N(\beta_b)$ denotes the most recently visited bus stop and $t_b \in T_{\beta_b}(n_b)$ is the time at which the bus b, during trip β_b, arrived at node n_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 *individual state of a traveler* is represented by (n, t), where $n \in N$ and $t \in T$, and denotes the node and time where a traveler is present in the network. The *system state space* S is a subset of the Cartesian product of the individual states of buses, the set of nodes, and the time period of interest, i.e., $S \subseteq \mathbb{S}$ where $\mathbb{S} = (\times_{b \in B} S_b) \times N \times T$. The construction of the set of individual states S_b and the system state space S will be discussed later. The state of the system at the instant the traveler departs from the origin is called the *initial state* or the *current state*. Table 1 lists the symbols used to describe the ATR problem.

Table 1: List of symbols

Symbol	*Description*
B	Set of buses (indexed by b)
R	Set of routes (indexed by r)
I_b	Itinerary of a bus b, which is the set of trips made by the bus (trips are indexed by β_b)
$N(\beta_b) \subseteq N$	Set of nodes visited by bus b in trip β_b
$A(\beta_b) \subseteq A_r$	Set of arcs included in trip β_b
$T_{\beta_b}(n) \subseteq T$	Set of times at which bus b reaches node n, where $n \in N(\beta_b)$
$f_{\beta_b}(n)$	Earliest possible time at which bus b reaches node n, where $n \in N(\beta_b)$
$l_{\beta_b}(n)$	Latest possible time at which bus b reaches node n, where $n \in N(\beta_b)$
$C_{\beta_b}(a)$	Random variable representing the travel time on arc a, where $a \in A(\beta_b)$
$\Omega_{\beta_b}(a)$	Support of travel time on arc a, where $a \in A(\beta_b)$
$c^{\omega}_{\beta_b}(a)$	ω^{th} element of $\Omega_{\beta_b}(a)$ (without loss of generality, assume that $c^{\omega}_{\beta_b}(a)$'s are arranged in increasing order, i.e., $c^{1}_{\beta_b}(a)$ is the least possible travel time on arc a)
w_{ij}	Walking travel time on arc (i, j), where $(i, j) \in A_w$ and $i \neq j$ (it is set to the cost of the path having the least walking time in the physical roadway network)
$\tilde{n}(s_b)$	Node associated with state s_b, where $b \in B$
$\tilde{t}(s_b)$	Time associated with state s_b, where $b \in B$
$order_b(s_b)$	Number of nodes visited by bus b between (being present in) the current state and state s_b (if the individual state of a bus b in the current state vector is s_b, $order_b(s_b) = 1$)

2.2 Assumptions

Given below is a list of assumptions used in this paper. 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 the algorithms and preprocessing methods discussed in later sections.

1. All buses have unlimited capacity.

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

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

4. Printed schedules for trips along a route in R and the bus to which each route trip is assigned is

assumed to be known. This information is used to construct the itinerary of each bus in B.

5. Each bus can serve multiple routes. However, a bus can serve route r_1 and subsequently route r_2 only if the destination of r_1 is the origin of r_2.
6. The time taken by buses during boarding and egress of passengers is neglected (this may be incorporated by adjusting the arc travel times if required).
7. Buses begin and end trips at a garage (represented by a node labeled 0) i.e., a bus in the network is always assumed to serve a route. This assumption is not restrictive since a bus that leaves the network and returns later may be considered as a new bus.
8. Travel times on transit arcs are independent random variables with finite support whose distributions are known. It is also assumed that the least possible travel time on a transit arc (i, j) is the difference between the scheduled arrival times at nodes j and i. In other words, buses are assumed to never arrive before the scheduled time. We make this assumption because drivers are typically instructed to wait at bus stops if they are early.
9. The travel time distribution on a transit arc traversed by a bus on a particular trip remains the same for every possible arrival time at its tail node. While the methods developed in this paper can be applied to cases in which the probability mass functions vary with time, this assumption makes it easier to illustrate the construction of the individual states of buses and the algorithms used for preprocessing.
10. The first trip made by a bus starts on time and buses begin new trips on schedule if they can by adjusting the slack between trips. Hence, if a bus arrives at or after the scheduled departure time at the origin of its next trip, it is assumed to proceed immediately without waiting.

Assumptions 1 and 6 are common to Hickman (1994), and Hickman and Bernstein (1997). While these authors assume that the slack in schedules is utilized even if a bus is delayed, we use Assumption 10 instead as we feel it is more practical. Assumptions 2, 8, and 9 are similar to the ones used in Miller-Hooks (2001), and Miller-Hooks and Mahmassani (2000).

2.3 Constructing the 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 of $C_{\beta_b}(a)$'s. Note that at t_O, buses are either in the network or in the garage. By convention, we assume that the state of a bus that is yet to enter the network is represented by the node-time pair $(0, -1)$ and a bus in the garage that left the network is represented by $(0, -2)$.

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. When route changes occur, the cost of the arc connecting the destination of one route and the origin of the other is modeled to reflect Assumption 10. If successive routes served by a bus are such that the destination of the first route is the origin of the next route, by construction, we have a copy of the node for each route and the arc connecting these nodes is used to model the slack. Figure 2 illustrates the process of construction of the individual states of a bus in a manner consistent with the assumed pmfs.

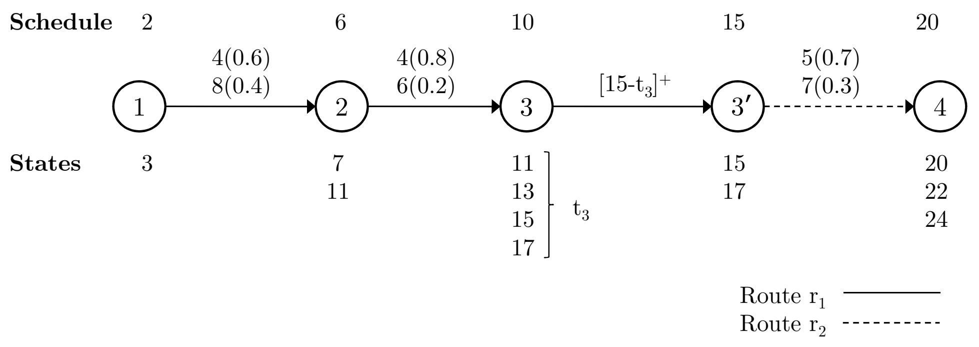


Figure 2: Individual states of a bus

Consider a bus b that was scheduled to arrive at stop 1 at time $t = 2$. Suppose that the current state vector indicates that the bus was last seen at stop 1 at time $t = 3$. The times associated with individual states are listed below the nodes in Figure 2. The values on the arcs indicate travel times and the corresponding probabilities are shown in parentheses. Since the travel time on arc (1,2) is either 4 or 8, the possible individual states at node 2 are $(2, 7)$ and $(2, 11)$. Proceeding similarly, we find the states at node 3 for each possible arrival time at node 2. Let node 3 be the destination of route r_1 and the origin of route r_2 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 $[15 - t_3]^+ = \max\{15 - t_3, 0\}$. 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 17, it proceeds immediately to serve the next route. As defined earlier, the states of a bus are assumed to be spatially ordered based on the number of nodes visited by it from its initial state, for instance, $order_b\big((1, 3)\big) = 1$. Likewise, *order* of the states at nodes 2, 3, $3'$, and 4 equals 2, 3, 4, and 5 respectively. Note that this procedure of constructing individual states holds even in the absence of Assumption 9. For instance, if the travel time on arc (2,3) is 4 or 16 when the bus arrives at node 2 at $t = 7$ and is 5 or 7 when it arrives at $t = 11$, then the individual states at node 3 are (3,11), (3,23), (3,16), and (3,18).

If a bus is in the garage at t_O, we append a node 0 before its first trip. For example, suppose a traveler departs from his/her origin at $t = 3$, and a bus b is scheduled to begin its trip 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. This network transformation is necessary for constructing the system state space.

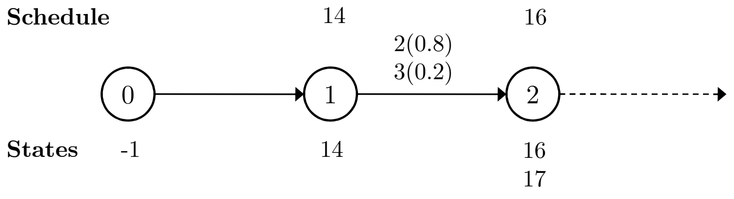


Figure 3: Individual states of a bus in the garage

2.4 Formal definition of the ATR problem

The ATR problem is formally defined as follows: Given a stochastic transit network (in which the transit travel times are time-dependent and random with known distributions), the initial state of the system, and a destination D; an adaptive policy which minimizes the total expected travel time is sought, subject to a constraint that D is reached within a threshold λ_D with probability 1 (*w.p.*1). While it is commonly assumed that route choices in stochastic networks are purely governed by the expected cost of travel; in minimizing this objective, a traveler might deviate significantly from an *a priori* path, or cycle multiple times before reaching the destination (see Hickman (1994) for a paradoxical example in which an adaptive policy that minimizes the expected cost may result in longer travel times along sample paths when compared with the optimal *a prori* strategy). Hence, we introduce an arrival time constraint that can help reduce such phenomena by modeling the extent of risk a traveler is willing to take.

As different individuals have different risk attitudes, the value of λ_D varies across travelers and is hence difficult to define. For a traveler with a hard arrival time constraint (say the traveler has to board a flight), the arrival time itself can be used as λ_D. In other cases, one could think along the lines of guaranteed returns to define λ_D, i.e., if a traveler has the option of choosing a strategy that ensures that D is reached within λ_D, he/she might not be inclined to follow a strategy which possibly takes longer to reach D. A naive value of λ_D is the shortest walking time between the origin and the destination. While computing this value is easy, it may be very high for trips spanning long distances and is thus impractical. In this paper, λ_D is defined (in Section 3.1) using the earliest time by which a traveler is guaranteed to reach D. Although we might find a policy that results in lower expected travel time in the absence of such a constraint, incorporating it improves realism and further helps reduce the state space.

Let us now mathematically formulate the ATR problem as an MDP. Given the *state space* S, let $x(s)$ and $X(s)$ denote an action and the set of available actions (*action/decision space*) at state $s \in S$ respectively. At each state, recognize that a traveler can wait or walk, or board a bus (if present at the stop). Therefore, let $X(s) = X_w(s) \cup X_r(s)$, where $X_w(s)$ and $X_r(s)$ comprises the waiting/walking and transit options available at state s respectively. While the cost of waiting/walking is deterministic, the travel time incurred by choosing a transit arc is random and is defined by the distributions of $C_{\beta_b}(a)$'s. In general, let $\tilde{\xi}_{x(s)}$ be the random *one-step cost* associated with decision $x(s)$. Further, let $\xi_{x(s)}$ and $\Xi_{x(s)}$ denote a generic value and the support of $\tilde{\xi}_{x(s)}$ respectively. Suppose $\mathbb{P}\big[s'|(s, x(s), \xi_{x(s)})\big]$ represents the *transition functions*, which provide the probability of ending up in state s' assuming that the decision maker incurs a cost of $\xi_{x(s)}$ by choosing $x(s)$ at state s. The *value function* at state s, denoted by $V(s)$, is the optimal expected cost of reaching D from state s. The value function for all states in which $n = D$ and $t = \lambda_D$ (where (n, t) is the individual state of the traveler) is set to 0 and the value of all other states is initialized at ∞. Using the notation and definitions described so far, the Bellman equation can be expressed as (1) and in order to solve for the optimal values, one can use backward induction.

$$V(s) = \min_{x(s) \in X(s)} \left[\mathbb{E}_{\Xi_{x(s)}} \left[\tilde{\xi}_{x(s)} + \sum_{s' \in S} \mathbb{P}\big[s' | (s, x(s), \tilde{\xi}_{x(s)})\big] V(s') \right] \right] \tag{1}$$

Clearly, the number of individual states of a bus increases exponentially with the number of trips in its itinerary which, in turn, results in an exponential number of system states. Thus, the ATR problem exhibits what is widely referred to as the *curse of dimensionality* and solving it by direct application of Bellman's principle becomes extremely difficult unless we find ways to reduce the state space.

3 Preprocessing procedure

3.1 Preview of preprocessing methodology and Light Cones

The optimal policy is not necessarily influenced by all the 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 impact the optimal policy. Further, as 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 ATR problem. The preprocessing steps described in this section aim at identifying the buses and individual states that could affect travel between an OD pair and thereby reduce the state space substantially. These steps make use of solutions to variants of the time-dependent shortest path (TDSP) problem, which are useful in two ways. First, they provide a rationale for defining the value of λ_D. Second, they help develop an elimination procedure which is explained in Sections 3.2 and 3.3. More specifically, the following four problems are considered:

(a) *Earliest Origin-to-All TDSP (EOA):*

Assuming that a traveler departs from the origin at t_O, the EOA problem involves computation of labels (denoted by μ_n, where $n \in N$) that represent the earliest possible time at which the nodes in the network can be reached with positive probability ($w.p. > 0$).

(b) *Earliest All-to-Destination TDSP (EAD):*

Given that we depart from a node n at time t, this problem involves finding labels $\gamma_n(t)$ which specify the earliest we can reach the destination $w.p. > 0$.

(c) *Latest Origin-to-All TDSP (LOA):*

In this problem, given that a traveler is at the origin at t_O, we determine a label (denoted by λ_n, where $n \in N$) for each node which represents the earliest we can reach the node $w.p.1$.

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

Given an individual state of a bus s_b, the problem is to find $\eta(s_b)$, the earliest we can reach the destination $w.p.1$ assuming that the individual state of the traveler is same as that of the bus. The *all* in LAD refers to all individual states and should not be confused with the nodes in the network.

Algorithms for the above problems are presented in Appendix A and may be skipped by the reader without loss of continuity. Let us now briefly discuss how the optimal labels of these problems help discard the states of the buses that do not affect the optimal policy.

We first solve the LOA problem and set λ_D to the label of the destination. This not only reflects the assumptions made on travelers' risk attitudes, but also ensures that a feasible solution to the ATR problem exists. Note that the state space reduces with decrease in the arrival time threshold, but the methods for preprocessing are not specific to a particular value of λ_D. We then discard all individual states of a bus if the earliest time at which the bus reaches a node is greater than λ_D and reduce the size of T by resetting the value of T_{max} to $\max_{b \in B, s_b \in S_b} \tilde{t}(s_b)$. Although this step is not strictly necessary, it speeds up the computation of the other problems as their complexity depends on T_{max}. Next, we solve the EAD, LAD, and EOA problems, the purposes of which may be 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., northbound)? If yes, do all individual states of the bus play a role in finding the optimal strategy?

2. Suppose a traveler just missed a bus. Can we ignore it while populating the system states?

Consider the first question. A traveler heading south might reach the destination faster by transferring to another bus after boarding the northbound bus, thus making a case for including the northbound bus while constructing the system state space. In other words, a user may travel away from the destination in the process of minimizing his/her expected travel time. However, all individual states of the bus may not aid in finding the optimal strategy and using the EAD and LAD labels, we might be able to discard the individual states from which the destination cannot be reached within λ_D. Let us now address second question. Even if a traveler misses a bus, the missed bus may be caught at a later stop using some faster service. This line of thought encourages us to employ the EOA labels to find the earliest possible arrival times at stops along the bus route and check if we can board the bus at some individual state.

The above mentioned elimination procedure is inspired by the concept of light cones used in relativistic physics. A light cone is a flash of light from an event (E) that travels in spacetime and consists of 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. Light cones serve as a tool to understand causality; only the events that occur in the past light cone can possibly affect E, and E can possibly influence only the events in the future light cone.

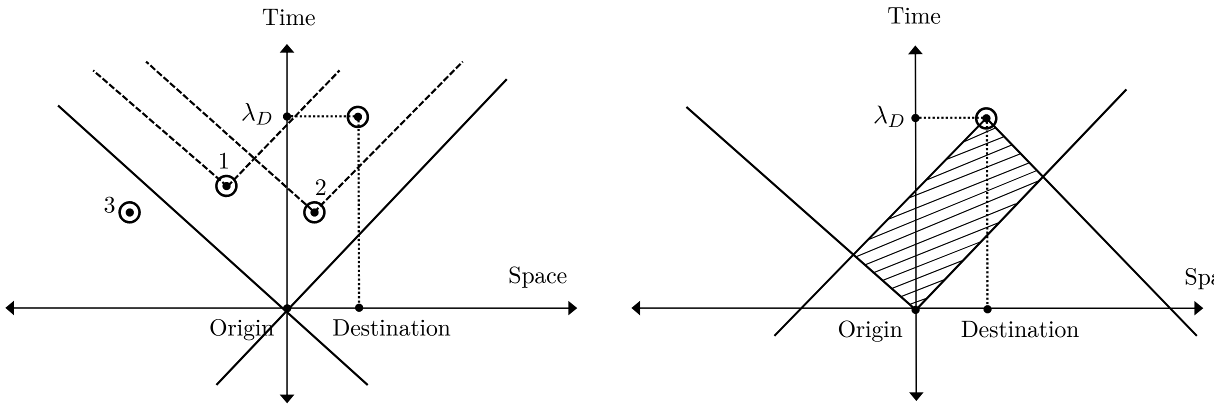


Figure 4: Light cones and the relevant states of a bus in the network

Consider the example shown in the left panel of Figure 4. Suppose that the network consists of a single bus, and a traveler starting from the origin at $t = 0$. For illustrative purposes, assume that the diagram representing how fast the traveler reaches the nodes in the network is a cone. The cone at the origin can be constructed using the EOA labels. Let points 1, 2, and 3 represent three states of the bus. State 3 lies outside the cone at the origin and hence cannot be reached. On the other hand, the traveler can reach 1 and 2 as they lie within the cone at the origin. Note that the point represented by the destination and λ_D lies inside the cone at 2, but falls outside the cone at 1 (this may be inferred using the EAD or LAD labels). Therefore, the only state which can potentially influence the optimal strategy is 2.

Another approach to visualize the elimination procedure is to construct a future cone from the origin at $t = 0$ and a past cone from the destination at $t = \lambda_D$ (see right panel of Figure 4). Individual states which lie in the intersection of the two cones can be expected to affect the optimal policy. The exact conditions under which an individual state can be ignored are examined in detail in the following sections.

3.2 Elimination of individual states using EAD/LAD labels

Since the traveler has to reach the destination within λ_D $w.p.1$, the EAD and LAD labels may be used to verify if boarding a bus at an individual state violates the arrival time constraint. This is accomplished by defining indicator variables δ_{EAD} and δ_{LAD} as shown below.

$$\delta_{EAD}(s_b) = \begin{cases} 1 & \text{if } \gamma_{\tilde{n}(s_b)}\left(\tilde{t}(s_b)\right) > \lambda_D \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\delta_{LAD}(s_b) = \begin{cases} 1 & \text{if } \eta(s_b) > \lambda_D \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

It is obvious that $\delta_{EAD}(s_b) = 1 \Rightarrow \delta_{LAD}(s_b) = 1$ and $\delta_{LAD}(s_b) = 0 \Rightarrow \delta_{EAD}(s_b) = 0$. When $\delta_{LAD}(s_b) = 1$ and $\delta_{EAD}(s_b) = 0$, a traveler at $\left(\tilde{n}(s_b), \tilde{t}(s_b)\right)$ cannot reach the destination within λ_D $w.p.1$. But since $\delta_{EAD}(s_b) = 0$, he/she can reach the destination before λ_D for some particular system state(s). Under such circumstances, based on the system state, the optimal policy might direct the traveler to catch the bus if it foresees that taking the bus does not violate the arrival time constraint and might suggest a different action otherwise. Hence, using the LAD labels one might eliminate more states than required. Though we present the results of preprocessing based on the LAD labels, it is beyond the scope of this paper to explore the trade-off between the computational advantages of dealing with a smaller state space obtained using the LAD labels and the value of the objective.

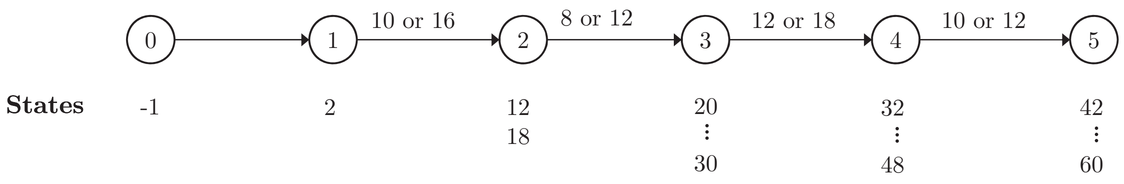


Figure 5: Elimination of the individual states of a bus

Let us study the elimination process using a bus whose individual states are shown in Figure 5. Table 2 contains the corresponding δ values of these states. Assume that $\lambda_D = 40$. Therefore, the states beyond node 4 can be discarded. We first discuss the elimination methods using the EAD labels and later extend it using the LAD labels. Let the individual states of the bus be represented as an acyclic network as shown in Figure 6. A node in this network denotes an individual state and arcs are used to connect adjacent states. Let the states that can be reached from an individual state and all the states from which a particular state can be reached be referred to as *descendant* and *ancestor* states respectively. For example, in Figure 6, the descendants of state $(2, 12)$ are (3,20), (3,24), (4,32), (4,38), (4,36), and (4,42); and the ancestors of state $(3, 20)$ are (2,12) and (1,2). We try and fathom the nodes in this network using the δ values, and hence one could think of this network as being similar to a branch and bound tree.

Table 2: δ_{EAD} and δ_{LAD} values for the individual states of the bus

State (s_b)	Node ($\tilde{n}(s_b)$)	Time ($\tilde{t}(s_b)$)	δ_{EAD}	δ_{LAD}
(1, 2)	1	2	0	1
(2, 12)	2	12	0	1
(2, 18)	2	18	1	1
(3, 20)	3	20	0	1
(3, 24)	3	24	0	1
(3, 26)	3	26	1	1
(3, 30)	3	30	1	1
(4, 32)	4	32	0	0
(4, 36)	4	36	0	1
(4, 38)	4	38	1	1
(4, 42)	4	42	1	1
(4, 44)	4	44	1	1
(4, 48)	4	48	1	1

For the sake of illustration, a few states have been replicated (indicated by dashed connectors) in Figure 6. All states with $\delta_{EAD} = 1$ are marked in grey. Note that if $\delta_{EAD} = 1$ for a particular state, we cannot reach the destination within λ_D with positive probability from all descendant states. The elimination of individual states is divided into the following two phases.

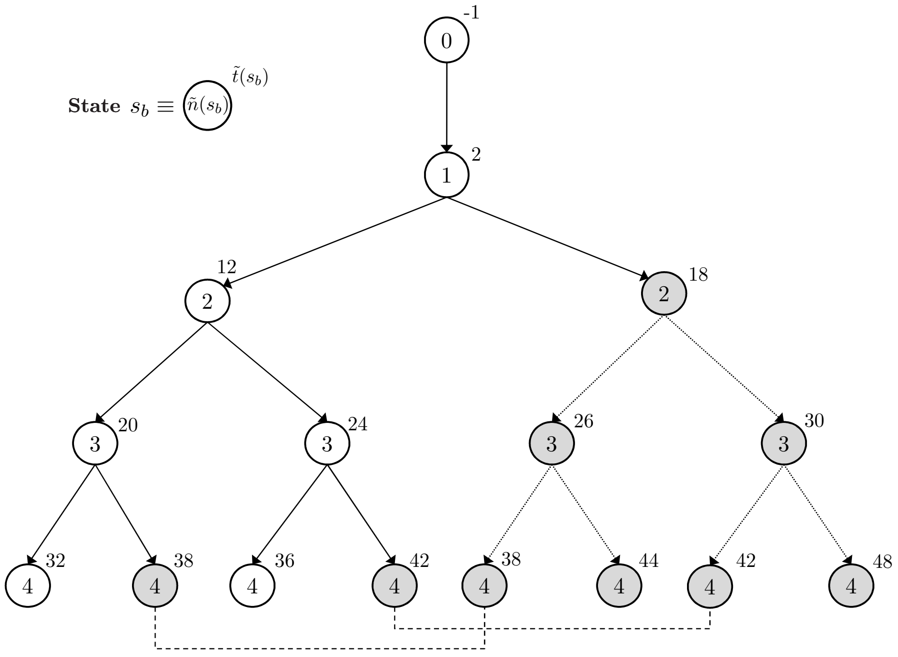


Figure 6: Acyclic network of individual states marked using EAD labels

Phase I:

From Figure 6, observe that if the bus at state (1,2) takes 16 minutes to travel to node 2, it certainly does not influence the optimal strategy in which case we may assume that it exits the network. This feature is modeled 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 dotted lines in Figure 6) and then delete all marked states. Arcs orphaned due to the removal of marked states are then connected to the 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 $(0, -2)$. The resulting network of individual states is shown to the left in Figure 7.

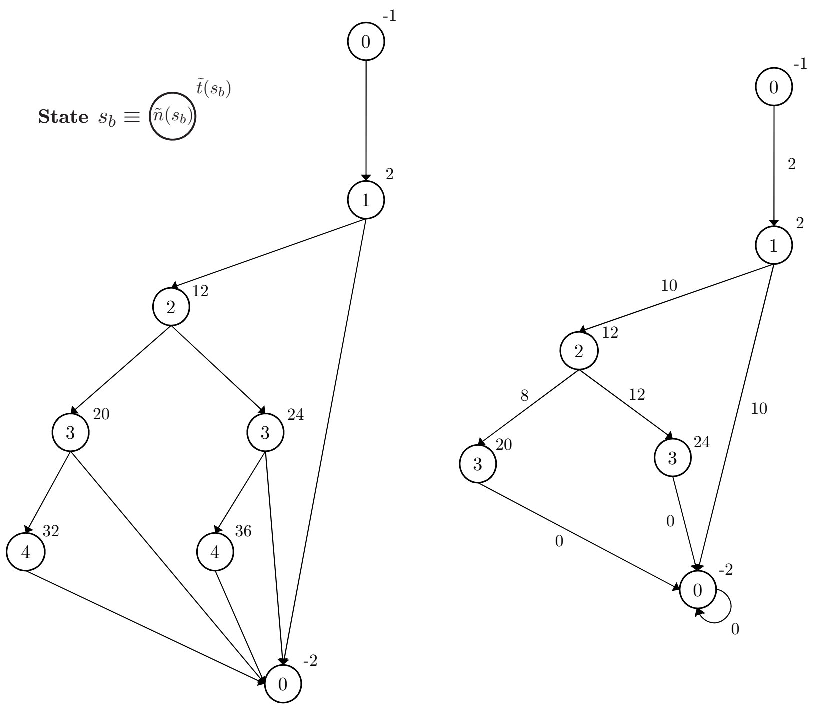


Figure 7: Phase I (left) and Phase II (right) elimination of individual states

Phase II:

Suppose a traveler is about to board the bus at node 3 at $t = 20$. Although the destination can be reached within λ_D $w.p. > 0$, the arrival time constraint may be violated if the bus takes 18 minutes to reach node 4. Hence, the state (4,32) can be eliminated and the bus may be assumed to proceed immediately from state (3,20) to the garage. The state (4,36) can also be eliminated using a similar argument.

In general, assuming that y_{max} represents the largest *order* among states in the acyclic network obtained from Phase I, 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 a state with *order* y_{max} is connected to $(0, -2)$, 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. The resulting set of states for the example under consideration is shown to the right in Figure 7 (the relevance of the values on the arcs connecting individual states is explained later in Section 4.1). We will henceforth refer to this figure as the *transition diagram of individual states.* Note that a loop connecting state $(0, -2)$ to itself has been added in order to ensure that the bus remains out of the network once it reaches the garage.

So far, we have discussed the use of the EAD labels in eliminating individual states. In a similar vein, a stronger, but approximate, elimination procedure can be developed using the LAD labels. 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 the Phase I and Phase II techniques to eliminate individual states. Figure 8 shows the acyclic network in which states have been marked using the δ_{LAD} values and the individual states resulting from Phase I. As mentioned 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 are eliminated, and hence the bus can be excluded while populating the system state space.

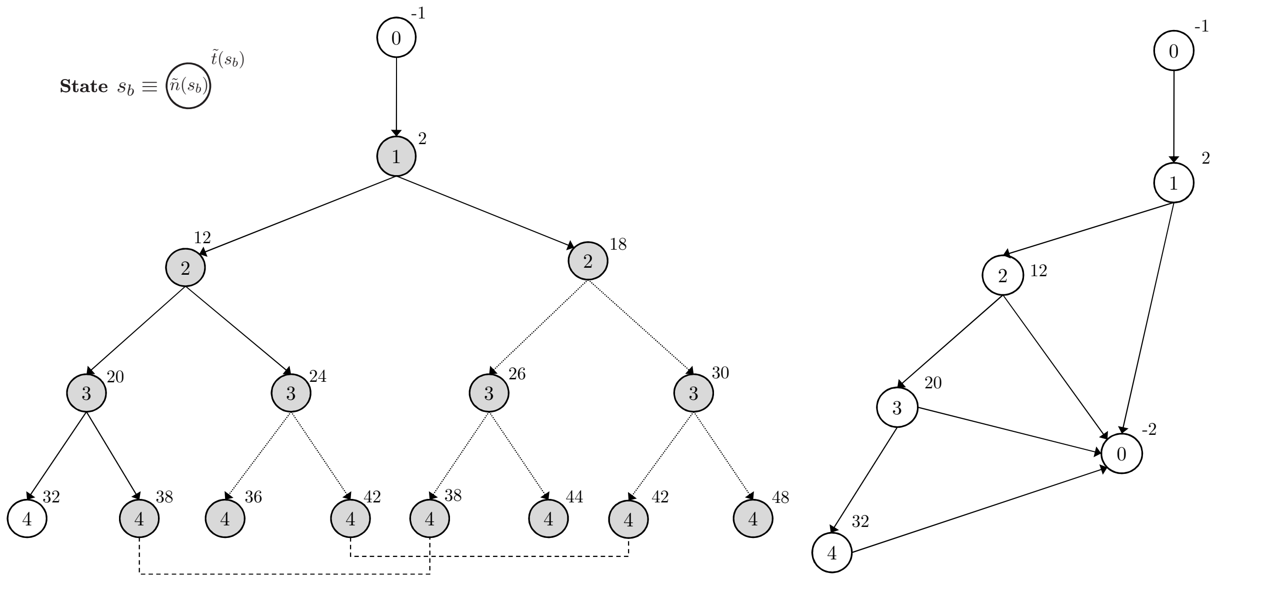


Figure 8: Acyclic network of individual states marked using LAD labels (left) and the individual states from Phase I (right)

Effect of elimination on the labels and δ values:

The elimination procedure was carried out independently for each bus without regard to its impact on the labels. However, doing so does not affect the validity of the procedure. Suppose we eliminate states using the EAD labels. In Phase I, if a particular state of a bus is eliminated, we know that by boarding the bus at that state, the destination cannot be reached within λ_D. If that state was used in finding the optimal EAD label of another individual state (of the same or of a different bus), δ_{EAD} of the latter state would be 1. Thus, eliminating the former state might increase the EAD labels but the δ values remain unaffected. A similar reasoning holds if the LAD labels are used to reduce the individual state space.

Now consider the Phase II of the elimination process. Since unmarked states may be eliminated in this phase, the δ 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 left panel of Figure 7, removing state (4,32) (and the arc between (3,20) and (4,32)) may preclude the possibility of reaching the destination within λ_D *w.p.* > 0 from some or all of its ancestors, or from the states of a different bus. Thus, we may iterate between the calculation of the EAD labels and the elimination phases until no more states are excluded. Although we have not investigated the computational advantages of doing so, we believe that this iterative procedure can further reduce the size of the individual state space. If the LAD labels are used to eliminate individual states, it is easy to see that the δ_{LAD} values do not undergo any change after Phase II.

3.3 Elimination of individual states using EOA labels

If the latest time at which a bus arrives at a node is less than its EOA label, 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:

$$\delta_{EOA}(s_b) = \begin{cases} 1 & \text{if } \mu_{\tilde{n}(s_b)} > \tilde{t}(s_b) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Consider the example in Figure 5 to illustrate the use of the δ_{EOA} variable. Let the vector [15 25 35 50] represent the EOA labels of nodes 1, 2, 3, and 4 (recall that the states at node 5 were discarded). The δ_{EOA} value of every state is 1, and hence we can ignore the bus while finding the optimal policy. Now consider another scenario in which the EOA labels are [15 25 30 42]. Clearly, we cannot reach nodes 1 and 2 before $t = 2$ and $t = 18$ respectively, but since the bus can be caught at node 3 at $t = 30$, it can be boarded at any node along the remainder of its journey with positive probability. However, the states at nodes 1 and 2 cannot be eliminated as the travel time realizations on arcs (1,2) and (2,3) shed more light on the state of the bus at subsequent nodes.

In such situations, we can completely ignore a bus if, for each individual state, either δ_{EOA} or δ_{EAD} is 1. Intuitively, this condition 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 λ_D. A stronger, but approximate, scheme for elimination can be similarly formulated using the values of δ_{LAD} instead of the δ_{EAD} values.

3.4 Remarks on the elimination procedure

The elimination methods descried earlier are not completely tight, in the sense that they do not fully eliminate the buses/states that do not affect the optimal policy. Consider the example shown in Figure 9. Assume that at $t = 0$, a traveler at node 1 is headed towards node 4. Also let two buses, one on each route, start from their respective origins at $t = 0$. It is easy to see that the value of λ_D is 20. Since the destination can be reached within λ_D (*w.p.* > 0 and *w.p.*1) from all individual states of the two buses, no states are eliminated (irrespective of whether the EAD or LAD labels are used). However, in trying to board the bus on route r_2 (which, for some particular pmfs, might result in lower expected cost when compared with boarding the bus on route r_1), we risk reaching the destination after $t = 20$. Although we evade this issue by ignoring the walking arc between nodes 3 and 4 (explained later in Section 4.2), we still have to deal with the redundant states of the bus on route r_2.

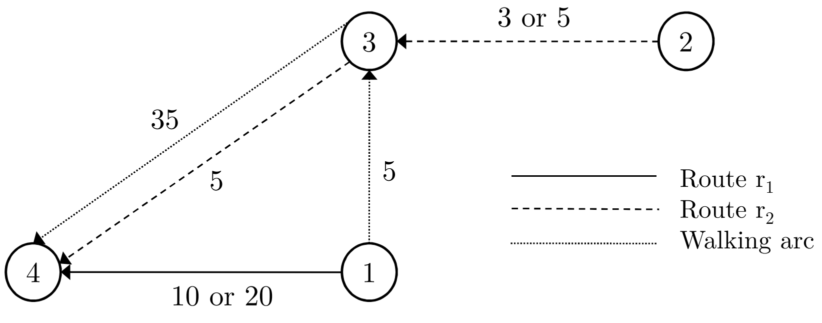


Figure 9: Drawbacks of the preprocessing techniques

4 Dynamic programming framework for the ATR problem

In this section, we define the elements of the MDP introduced in Section 2.4. Specifically, the following components of the dynamic program are discussed in detail: the state space, action space, and transition probabilities.

4.1 State space

Recall that the state space S is a subset of $\mathbb{S}$, where $\mathbb{S} = (\times_{b \in B} S_b) \times N \times T$. Assume that the set of system states at time $t \in T$ is denoted by $\mathcal{S}_t$. Note that the preprocessing methods cut down the size of the system state space by reducing the sizes of B and $\{S_b\}_{b \in B}$. Further, we redefine the time period of interest as $T = \{t_O, (t_O + 1), \ldots, \lambda_D\}$. Let the individual states $(0, -1)$ and $(0, -2)$ be referred to as the *source* and *sink* respectively. Additionally, let t^b_{source} and t^b_{sink} denote the time at which bus b enters the network and the earliest possible time at which it reaches the garage respectively. If the bus is already present in the network, the source is irrelevant and t^b_{source} is set to 0. In the transition diagram of individual states, let the cost of an arc from state s_b to s'_b, where $s_b \in S_b$ and $s'_b \in \hat{\Gamma}(s_b)$(the set of successor/downstream states, i.e., states directly connected to s_b), be defined as follows:

$$\alpha(s_b, s'_b) = \begin{cases} 0 & \text{if } \hat{\Gamma}(s_b) = \{(0,-2)\} \\ \tilde{t}(s'_b) - \tilde{t}(s_b) & \text{if neither } s_b \text{ or } s'_b \text{ represents the source or sink} \\ \max\limits_{s''_b \in \hat{\Gamma}(s_b) \setminus \{s'_b\}} \alpha(s_b, s''_b) & \text{if } |\hat{\Gamma}(s_b)| \geq 2 \text{ and } s'_b = (0,-2) \\ t^b_{source} & \text{if } (0,-1) \in S_b \text{ and } s_b = (0,-1) \end{cases} \tag{5}$$

Let us revisit the individual states shown in Figure 7. The numbers on the arcs in the transition diagram indicate the α values. The first and second cases in (5) deal with situations in which the α values represent the time taken by the bus to travel between s_b and s'_b. Examples of these cases include the arcs connecting states (3,20) and $(0, -2)$, and (2,12) and (3,20). In the third case, s_b is assumed to be directly connected to the sink and at least another individual state. For instance, consider the arc between states (1,2) and $(0, -2)$. Recall that if the bus takes 16 minutes to travel between stops 1 and 2, the destination cannot be reached within λ_D, and hence we that assume that the bus leaves the network. By convention (the reason for which is explained in Section 4.3), the cost of this arc is set to the maximum among the costs of all other outgoing arcs from state (1,2). The fourth case is again a convention that aids in the construction of the set of system states. Associated with each arc is a parameter $\pi(s_b, s'_b)$ that represents the probability with which state s'_b can be reached from s_b. This may be calculated using the distributions of $C_{\beta_b}(a)$'s. However, if one of the states is the sink or the source, we write $\pi(s_b, s'_b)$ as follows:

$$\pi(s_b, s'_b) = \begin{cases} 1 & \text{if } \hat{\Gamma}(s_b) = \{(0,-2)\} \text{ or if } (0,-1) \in S_b \text{ and } s_b = (0,-1) \\ 1 - \sum\limits_{s''_b \in \hat{\Gamma}(s_b) \setminus \{s'_b\}} \pi(s_b, s''_b) & \text{if } |\hat{\Gamma}(s_b)| \geq 2 \text{ and } s'_b = (0,-2) \end{cases} \tag{6}$$

Notice that at $t = 15$, the bus in Figure 7 cannot be at states (1,2), (3,20), and (3,24). While it cannot be at the latter two states since they represent points in future, a bus at state (1,2) would have advanced to one of its successors by $t = 12$. Hence, the bus can only be at either (2,12) or $(0, -2)$. More generally, the procedure in Algorithm 1 can be used to mark individual states at which a bus b might possibly be present at time $t \in T$.

Algorithm 1 *Marking individual states*

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

    if $t \geq t^b_{sink}$ then                                               ▷ Condition IV
        Mark sink
    end if
end if
```

The algorithm first compares t with the time at which the bus enters the network. If $t < t^b_{source}$, the bus is still at the garage. Hence, we mark the source and terminate. For example, at $t = 1$, the bus in Figure 7 can only be present at $(0, -1)$. Next, we scan all states except the source and sink, and the states from which the bus immediately proceeds to the garage. If t coincides with the time associated with an individual state, we mark it using condition II. Condition III ensures that a state in future is never marked, and an individual state is marked only if it does not advance to one of its descendants $w.p.1$. Finally, if $t \geq t^b_{sink}$, the bus could possibly be present at the garage, and hence we mark the sink using condition IV. Although the states from which the bus immediately proceeds to the garage (e.g., (3,20) and (3,24)) are never marked, they are not excluded from the transition diagram of individual states as they help define the action space and the transition probabilities.

In order to construct the set S, we first use Algorithm 1 to mark the states of each bus at a given time t. We then define $\mathcal{S}_t$ as the Cartesian product of the collection of sets representing the marked individual states of buses, N (additionally, using the EAD labels, one could exclude the nodes from which D cannot be reached $w.p. > 0$), and $\{t\}$. Finally, we repeat this procedure for all $t \in T$, and set $S = \bigcup_{t \in T} \mathcal{S}_t$.

4.2 Action space

Given a system state $s = (s_{b_1}, s_{b_2}, \ldots, s_{b_{|B|}}, n, t)$, the action space $X(s)$ is the set of actions available to a traveler at (n, t). We represent a particular action $x(s)$ using the pair $(arc, mode)$. The value of $mode$ is set to 0 if the traveler chooses a walking arc, and equals b if the traveler decides to board bus b. Note that $X(s) = X_w(s) \cup X_r(s)$, where $X_w(s)$ and $X_r(s)$ comprises the waiting/walking and transit options at state s respectively. $X_w(s)$ may be defined as follows:

$$X_w(s) = \big\{\big((i,j),0\big) : (i,j) \in A_w,\, i = n,\, \gamma_j(t + w_{ij}) \leq \lambda_D\big\} \tag{7}$$

The condition $i = n$ ensures that the traveler can only select the waiting/walking arcs that emanate from node n. Additionally, we compare the EAD label of node j at time $(t + w_{ij})$ with λ_D to make sure that choosing arc (i, j) does not violate the arrival time constraint. For example, as seen in Section 3.4, we exclude the walking arc (3,4) from the action space of all states in which the traveler is at node 3. Using the walking arcs in A_w is limiting in the sense that a traveler is expected to traverse an entire arc before making his/her next decision. In practice, using the latest information on buses in the network, one might choose an alternate strategy while walking between a pair of nodes. This feature can be modeled using a denser network of walking arcs, however, at the cost of increasing the size of the action space.

Now consider the transit options available at state s. Clearly, a transit arc can be traversed only if the individual state of the traveler is same as that of a bus in the network. Hence, we write $X_r(s)$ as follows:

$$X_r(s) = \Big\{\big((i,j),b\big) : (i,j) \in A_r,\, b \in B,\, i = n = \tilde{n}(s_b),\, t = \tilde{t}(s_b),\, (0,-2) \notin \hat{\Gamma}(s_b)\Big\} \tag{8}$$

In the above definition, conditions $n = \tilde{n}(s_b)$ and $t = \tilde{t}(s_b)$ imply that the traveler can board bus b. We also ensure that the traveler, after boarding bus b, can get off at a node in the network $w.p.1$ by verifying that the sink isn't a successor of s_b. For example, in the transition diagram in Figure 7, transit arc (2,3) can be chosen only if the traveler and the bus are at node 2 at $t = 12$.

4.3 Transition functions

Suppose a traveler at state s chooses action $x(s)$. The transition functions help describe the evolution of the system and are denoted by $\mathbb{P}\big[s'|(s, x(s), \xi_{x(s)})\big]$, where $s' \in S$ and $\xi_{x(s)} \in \Xi_{x(s)}$. To estimate these functions, we find the probabilities of the future states of each bus (i.e., $\mathbb{P}\big[s'_b|(s, x(s), \xi_{x(s)})\big]$, where $b \in B$ and $s'_b \in S_b$) separately by exploiting the independence of transit travel times (see Assumption 8). If $x(s)$ involves boarding a bus b or if a bus b is back in the garage, then calculating $\mathbb{P}\big[s'_b|(s, x(s), \xi_{x(s)})\big]$ is trivial; else we use the methods developed in this section to compute $\mathbb{P}\big[s'_b|(s, x(s), \xi_{x(s)})\big]$.

The knowledge of s_b and t (which may be obtained from the system state vector s) provides two useful pieces of information. First, we may infer that the bus has been traveling for $t - \tilde{t}(s_b)$ since it was last seen at a bus stop. Second, we can determine the set of "realized" states, which is defined as individual states the bus is guaranteed to not reach. Consider the following two scenarios: (i) suppose that at $t = 15$, if the bus in Figure 7 is at state (2,12), then none of the downstream states of (2,12) are realized; (ii) instead, if the state of the bus is (2,12) at $t = 21$, the bus would not have reached node 3 at time 20 $w.p.1$, and hence we can be certain that the state (3,20) is realized.

In the discussion that follows, let $\hat{s}_b$ denote a successor of s_b and $\hat{\hat{s}}_b$ represent a successor of $\hat{s}_b$. Also let t_e^b be the time elapsed since $\tilde{t}(s_b)$ (i.e., the bus is assumed to travel for t_e^b minutes from state s_b). In order to compute the probabilities of the future states of bus b, we first calculate $\Pr\left[s_b'|(s_b, t_e^b, \hat{s}_b)\right]$ (where $s_b' \in S_b$ and $\hat{s}_b \in \hat{\Gamma}(s_b)$), which denotes the probability of finding the bus at s_b' given that the set of individual states $\left\{\hat{s}_b' \in \hat{\Gamma}(s_b) : \alpha(s_b, \hat{s}_b') < \alpha(s_b, \hat{s}_b)\right\}$ are realized. We may then interpret $\mathbb{P}\left[s_b'|(s, x(s), \xi_{x(s)})\right]$ as the probability of finding the bus at s_b' in $\xi_{x(s)}$ minutes from time t given that it was last seen at some stop at $\tilde{t}(s_b)$, i.e., the time elapsed t_e^b is $t - \tilde{t}(s_b) + \xi_{x(s)}$. Assuming $\tilde{t}(s_b)$ is set to 0 if $s_b = (0, -1)$, we therefore write $\mathbb{P}\left[s_b'|(s, x(s), \xi_{x(s)})\right] = \Pr\left[s_b'|\left(s_b, (t - \tilde{t}(s_b) + \xi_{x(s)}), \hat{s}_b\right)\right]$, where $s_b \in S_b \backslash \{(0, -2)\}$, $s_b' \in S_b$ and $\hat{s}_b = \underset{\substack{\hat{s}_b' \in \hat{\Gamma}(s_b): \\ \alpha(s_b, \hat{s}_b') + \tilde{t}(s_b) > t}}{\arg\min} \left[\alpha(s_b, \hat{s}_b') + \tilde{t}(s_b) - t\right]$(ties are broken in favor of non-garage states).

In the example under consideration, let $\pi\big((2, 12), (3, 20)\big) = p$ and assume that the $\Pr[\,.\,]$ values for $t_e^b = 10$ are known. In Scenario (*i*), if the traveler chooses an arc of cost 7 (i.e., $\xi_{x(s)} = 7$), then $t - \tilde{t}(s_b) + \xi_{x(s)} = 10$ and since no downstream states are realized, we find the probability of future states using $\Pr\left[(0, -2)|\big((2, 12), 10, (3, 20)\big)\right] = p$ and $\Pr\left[(2, 12)|\big((2, 12), 10, (3, 20)\big)\right] = 1 - p$. Likewise, if the traveler in Scenario (*ii*) incurs a cost of 1 unit in choosing an action, we use $\Pr\left[(2, 12)|\big((2, 12), 10, (3, 24)\big)\right] = 1$ to calculate the probabilities of the future states of the bus.

Due to their repeated use in solving the MDP, one can calculate and store the values of $\Pr[\,.\,]$ for all t_e^b bounded by the largest possible travel time on any arc. We now briefly describe the process of finding these values using a backward induction type argument. Clearly, for all t_e^b and s_b such that $(0, -2) \in \hat{\Gamma}(s_b)$, $\Pr\left[s_b'|\big(s_b, t_e^b, (0, -2)\big)\right]$ is 1 if s_b' is the sink and is 0 otherwise. In all other cases, proceeding backwards (i.e., examining the states with the largest *order* first), we may write a recursive equation as follows:

$$\Pr\left[s_b'|(s_b, t_e^b, \hat{s}_b)\right] = \sum_{\substack{\hat{s}_b' \in \hat{\Gamma}(s_b): \\ \alpha(s_b, \hat{s}_b') \geq \alpha(s_b, \hat{s}_b)}} \left(\frac{\pi(s_b, \hat{s}_b')}{1 - \sum_{\substack{\hat{s}_b'' \in \hat{\Gamma}(s_b): \\ \alpha(s_b, \hat{s}_b'') < \alpha(s_b, \hat{s}_b)}} \pi(s_b, \hat{s}_b'')} \right) F(s_b', s_b, t_e^b, \hat{s}_b'), \qquad (9)$$

$$\text{where } F(s'_b, s_b, t_e^b, \hat{s}'_b) = \begin{cases} 0 & \text{if } \left(t_e^b - \alpha(s_b, \hat{s}'_b)\right) < 0 \text{ and } s'_b \neq s_b \\ 1 & \text{if } \left(t_e^b - \alpha(s_b, \hat{s}'_b)\right) < 0 \text{ and } s'_b = s_b \\ \Pr\left[s'_b | \left(\hat{s}'_b, (t_e^b - \alpha(s_b, \hat{s}'_b)), \hat{\hat{s}}_b\right)\right] & \text{if } \left(t_e^b - \alpha(s_b, \hat{s}'_b)\right) \geq 0 \end{cases} \tag{10}$$

and $\hat{\hat{s}}_b$ is a successor of $\hat{s}'_b$ such that $\alpha(\hat{s}'_b, \hat{\hat{s}}_b)$ has the lowest cost among all arcs from state $\hat{s}'_b$ (once again breaking ties in favor of non-garage states). Using the fact that the bus travels for t_e^b minutes from state s_b, we first find downstream states $(\hat{s}'_b)$ which are not realized (marked in grey in Figure 10). The posterior probabilities of reaching these states are then computed using the first expression in the summation in (9).

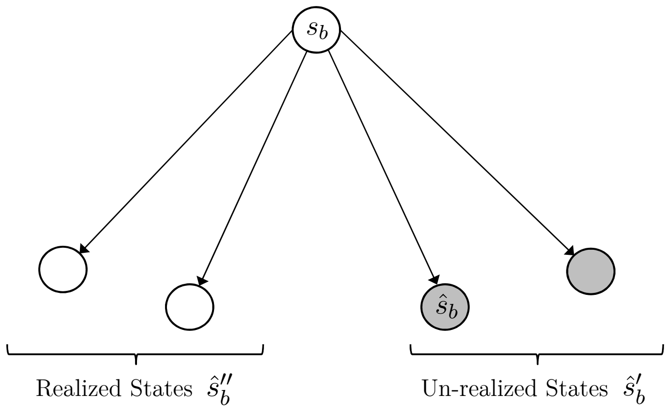

s_b
$\hat{\hat{s}}_b$
Realized States $\hat{s}''_b$
Un-realized States $\hat{s}'_b$

Figure 10: Calculating the transition probabilities

If the time elapsed is less than the cost of the arc from s_b to $\hat{s}'_b$, the bus would still be in state s_b, and hence we set the value of $F(.)$ in (10) to 1 if $s'_b = s_b$ and to 0 otherwise. Instead, if t_e^b is greater than or equal to the cost of the arc that connects s_b to $\hat{s}'_b$, the bus is assumed to travel for $t_e^b - \alpha(s_b, \hat{s}'_b)$ minutes from $\hat{s}'_b$ and using the state $\hat{\hat{s}}_b$, we ensure that none of the successor states of $\hat{s}'_b$ are realized (it is for this reason that ties are broken in favor of non-garage states). The probability of reaching s'_b is then obtained using the previously calculated value of $\Pr\left[s'_b | (\hat{s}'_b, t_e^b - \alpha(s_b, \hat{s}'_b), \hat{\hat{s}}_b)\right]$.

In order to illustrate the above procedure, we now derive the $\Pr[\,.\,]$ values used in Scenarios (i) and (ii).

$$\begin{aligned}
\Pr\big[(0,-2)|\big((2,12),10,(3,20)\big)\big] &= p\cdot F\big((0,-2),(2,12),10,(3,20)\big)+(1-p)\cdot F\big((0,-2),(2,12),10,(3,24)\big)\\
&= p\cdot \Pr\big[(0,-2)|\big((3,20),2,(0,-2)\big)\big]+(1-p)\cdot 0 = p\\
\Pr\big[(2,12)|\big((2,12),10,(3,20)\big)\big] &= p\cdot F\big((2,12),(2,12),10,(3,20)\big)+(1-p)\cdot F\big((2,12),(2,12),10,(3,24)\big)\\
&= p\cdot \Pr\big[(2,12)|\big((3,20),2,(0,-2)\big)\big]+(1-p)\cdot 1 = 1-p\\
\Pr\big[(2,12)|\big((2,12),10,(3,24)\big)\big] &= \frac{1-p}{1-p}\cdot F\big((2,12),(2,12),10,(3,24)\big)\\
&= 1\cdot 1 = 1
\end{aligned}$$

Before concluding this section, we explain the reason behind the third case in (5) using the following example. Suppose the bus in Figure 7 is at state (1,2). If $t_e^b = 10$, we expect the bus to be either at (2,12) or at $(0,-2)$, and thus we defined the cost of the arc between states (1,2) and $(0,-2)$ to be 10. Instead, if we set it to a value greater than 10, the expressions used to compute the transition functions would incorrectly imply that the future state of the bus is either $(2,12)$ or $(1,2)$.

5 Demonstration

In this section, we apply the state space reduction methods and the dynamic program to a small portion of the Austin transit network.

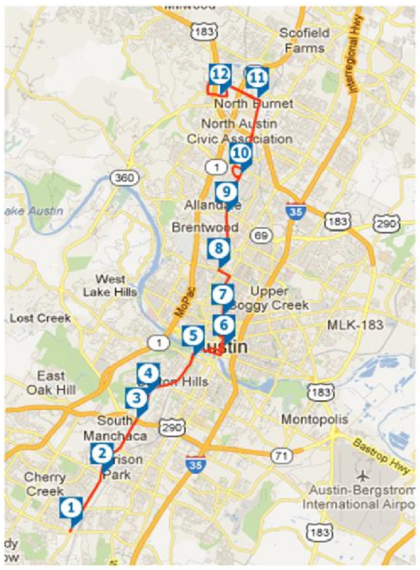
5.1 Network Description

Eight routes on the Austin transit network were chosen and the information available on the Capital Metropolitan Transportation Authority's website was used for this study. Table 3 shows a summary of the input data.

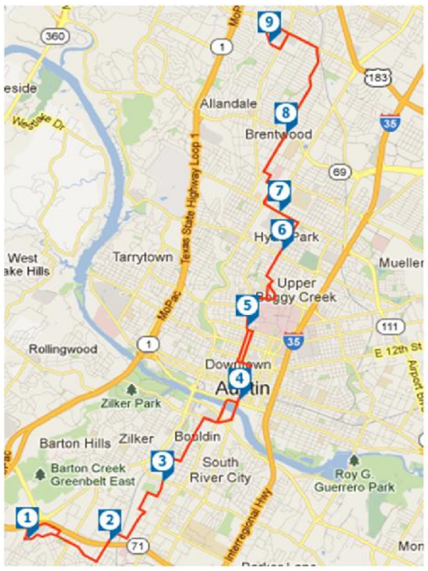
Table 3: Input Data

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

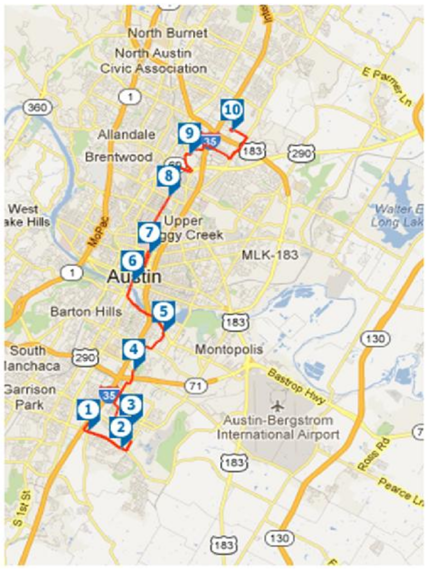
Figure 11 shows the routes and time points at which the scheduled arrival times for each trip were known. All trips that begin between 6:00 AM and 12:00 PM were included in the model. A total of 78 stops were considered, and the shortest walking time between each pair of nodes was obtained using the Google Distance Matrix API. The implementation was carried out in C++ (using the g++ compiler with −O3 optimization flags) on a Linux machine with a 4 core Intel Xeon processor (3.47 GHz) and 12 MB cache.



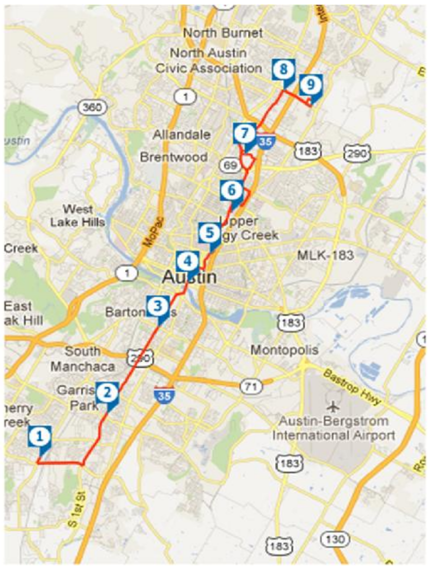
Route 3



Route 5



Route 7



Route 10

Figure 11: Routes (Source: http://www.capmetro.org/)

Itineraries were constructed for each bus by manually assigning trips to buses. In some cases, buses were assigned to trips along multiple routes (e.g., buses on route 10 also serve route 3). The current state vector and the pmfs of transit arc costs were randomly generated (the support size of travel times on all transit arcs was restricted to 2), and were used to construct the individual states of buses in the network.

5.2 Numerical results

Table 4 shows the number of individual states of buses after each stage of preprocessing for a problem instance in which the value of λ_D was found to be 48 minutes.

Table 4: Results of individual state space elimination

Bus ID	*Before elimination*	*Arrival time constraint*	*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

The arrival time constraint column indicates the number of states for which $\tilde{t}(s_b) \leq \lambda_D$ (buses that were completely eliminated using this criteria are not shown in the table). Although the results of the Phase I and II elimination methods are extremely appealing when examined in contrast with the initial number of individual states, it would only be fair to compare them with the size of the individual state space after imposing the arrival time constraint.

The plot in Figure 12 shows the logarithm of the cardinality of the Cartesian product of the individual state space (which is larger but comparable to the size of the actual state space) for various preprocessing steps. While the magnitude of the reduction in the state space is significant after Phase I, further reduction due to Phase II appears to be marginal. As expected, more states were eliminated using the LAD labels. However, we did not observe any reduction in the state space using the EOA labels.

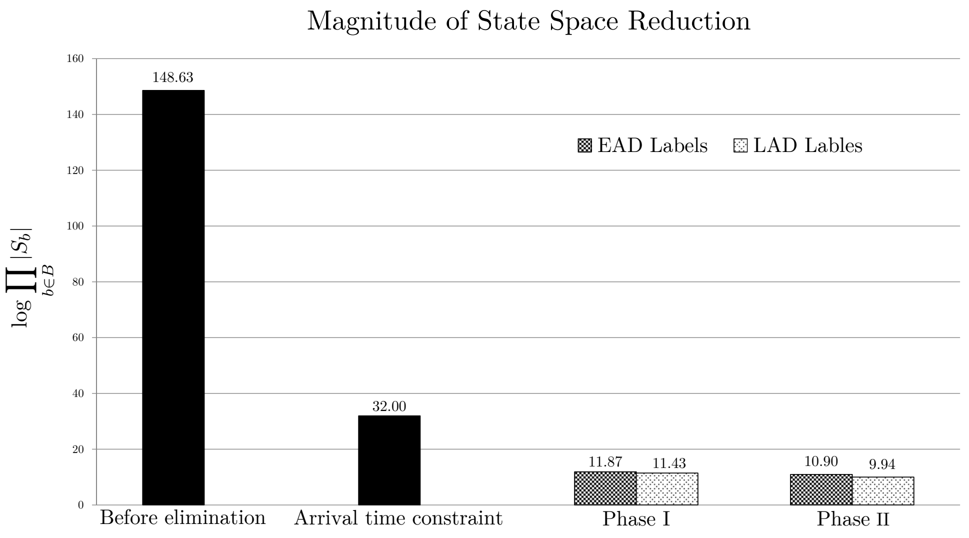


Figure 12: Reduction in the size of the individual state space

The magnitude of decrease in the size of the state space is primarily influenced by the EAD and LAD labels, which in turn are affected by factors such as the OD pair, the density of the transit network, and the pmfs of transit travel times. While the numerical results presented here roughly reflect the abilities of the preprocessing methods, it is difficult to predict the performance of these methods when applied to a different network. For instance, one might assume that having a denser network of buses would make it easier to reach the destination within λ_D from any individual state, and hence fewer states may be expected to be eliminated. However, in a dense transit network, the value of λ_D might be very low in the first place.

After preprocessing using the EAD labels, Algorithm 1 was employed to construct the system state space, the size of which was found to be about 9.37E+08. The action space was then populated as explained in Section 4.2. Since it is reasonable to expect some threshold on the distance a transit user might walk, we only considered walking arcs for which $w_{ij} \leq 15$. The transition probabilities were then determined using the $\Pr[\,.\,]$ values, and finally the value functions were computed using backward induction. Note that the dummy arcs used to model slack could have a cost 0, and hence a few system states may be equivalent to each other. Therefore, while implementing the backward induction algorithm, if the value of a state is updated, it is necessary to simultaneously update the values of all of its equivalent states. Table 5 shows the wall-clock times for preprocessing and the components of the dynamic program (the time taken for the transition functions step only includes the time spent in estimating the $\Pr[\,.\,]$ values). The results demonstrate that the state space reduction methods developed in this paper can improve the computational tractability of the ATR problem. Further, as almost all the steps involved in solving the ATR problem are parallelizable, one can expect near-linear speedup on shared memory systems.

Table 5: Wall-clock times in seconds for various stages of the ATR problem

Step	*Time (seconds)*
Preprocessing	3.603
System space	37.388
Action space	3.986
Transition functions	0.011
Backward Induction	393.707
Total (including I/O)	444.566

Of the 12 buses that were not eliminated during the preprocessing stage, only 2 were used in the optimal policy. A portion of the routes served by the two buses is shown in Figure 13. The origin and destination of the traveler are represented by nodes 1 and 5 respectively. The current state vector indicates that the individual state of the traveler and that of the bus on route 7 match, and that the bus on route 10 was last seen at node 1 a few minutes before t_O. Thus, the two *a priori* strategies available are to board the bus on route 7 and walk to the destination from node 4 or to wait for another bus that serves route 10, of which boarding the bus on route 7 is optimal and has an expected cost of 37.6 minutes.

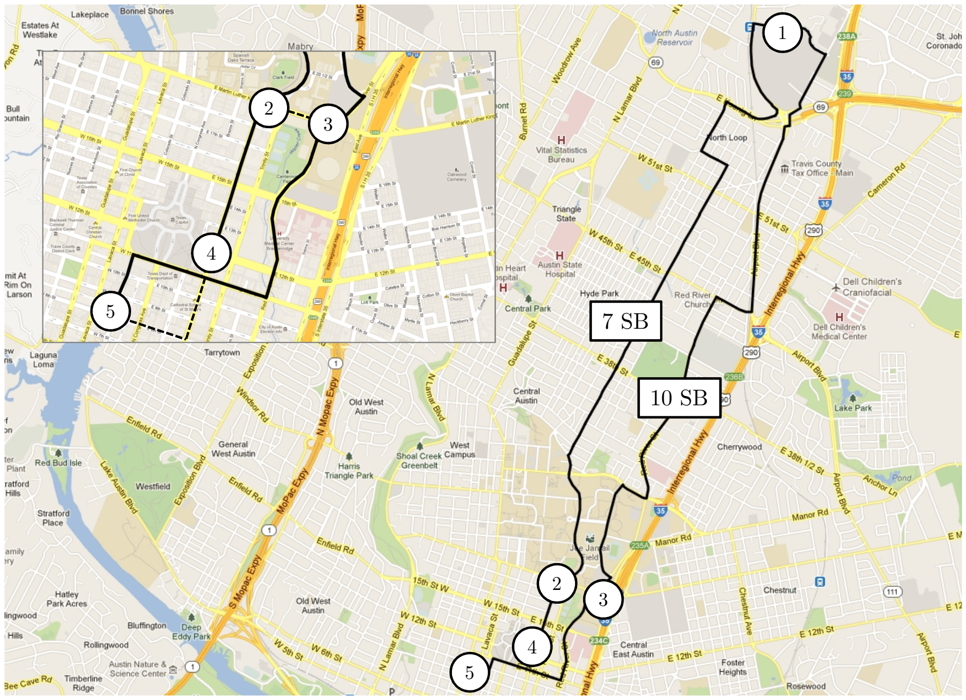


Figure 13: Routes used in the optimal policy (dashed lines represent walking arcs)

On the other hand, the adaptive strategy suggests that the traveler boards the bus on route 7 and, based on the information received while traveling, either gets off at node 4 or transfers to the bus on route 10 at node 3 (after choosing the walking arc (2,3)). The expected cost of the optimal adaptive strategy was found to be 37.5025 minutes, reiterating the fact that the adaptive framework defined in this paper outperforms the optimal *a priori* solution.

Since the probability distributions used in these calculations were randomly generated, no definite conclusions can be drawn on the magnitude of the benefits of using an adaptive strategy. In general the travel time savings depend on several factors such as the OD pair, the pmfs, and the density of the transit network. For instance, in the example in Figure 1, the expected travel time savings is 1.5 minutes, which amounts to a 25% decrease when compared with the expected cost of the optimal *a priori* strategy. Instead, if the travel time on arc (4,5) is either 1 or 4, the optimal cost of the *a priori* strategy and that of the adaptive solution are the same.

6 Conclusions

In this paper, an adaptive transit routing problem is discussed in which a traveler seeks a strategy that minimizes the expected cost of travel while ensuring that the destination is reached within a certain threshold. The problem can be easily extended to include other modes of transit such as rail by including train lines and trains in the set of routes and buses respectively. In practice, a major challenge in this problem involves working with a large state space. We therefore emphasized reducing the size of the state space by independent reduction of the individual state space of each bus because not all bus states influence the optimal strategy. This process was facilitated by a few algorithms and assumptions on the behavioral attitudes of travelers. A numerical experiment to illustrate these methods was also performed.

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. Major contributions of this paper include the state space reduction methods and the MDP formulation, which enable us to use a vast amount of real time information that can potentially result in lower expected travel times when compared with an *a priori* solution. Also, the bus based approach let us model strategies and the slack in schedules between trips in a much broader and realistic manner when compared with traditional line based approaches.

However, the proposed model is limited by the assumptions made, some of which are restrictive in nature. For instance, the transit travel times are assumed to be independent of each other. In reality, buses on routes that share arcs are likely to be affected by congestion in a similar manner. 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 a constraint on the number of transfers in the present model is difficult. Also, the effects of fares and capacity of buses have been ignored in the decision making process.

The study of the ATR problem presents some useful pointers for future research. While the preprocessing methods look attractive, an exact estimation of the optimal strategy may be hindered by the size of the state space in larger networks. Since there exists more room for reducing the state space as noted in Section 3.4, the combinatorial problem of selecting the set of buses that influence the optimal solution needs further exploration. If the dynamic program remains intractable after the preprocessing stage, it may be worthwhile to explore approximate dynamic programming (ADP) techniques to approximately estimate the value functions (see Powell (2007) for an in-depth discussion on ADP).

It is also of prime interest to determine the value of information or the magnitude of travel time savings of the proposed methods in comparison with *a priori* strategies or other existing adaptive approaches. Although the ATR problem remains to be examined in greater detail, this paper develops a sound theoretical framework and novel state space reduction techniques that contribute significantly to the study of adaptive routing in transit networks.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1157294. The authors are solely responsible for any errors or omissions in this research paper. Special thanks to two anonymous referees and the associate editor whose comments have greatly improved the content and organization of this paper.

Appendix A Algorithms for preprocessing

Variants of the TDSP problem defined in Section 3.1 are non-trivial due to the bus schedule structure and the possibility of transfers. In this appendix, examples and labeling algorithms for these problems are presented and discussed in detail. Let SE be a scan eligible list, $\Gamma(i)$ denote the set of nodes adjacent to node i (i.e., $j \in \Gamma(i) \Leftrightarrow (i,j) \in A$), and $\Gamma^{-1}(i)$ represent the set of nodes from which node i can be directly reached (i.e., $j \in \Gamma^{-1}(i) \Leftrightarrow (j,i) \in A$). In the discussion that follows, it is assumed that the reader is familiar with basic TDSP algorithms, a comprehensive summary of which can be found in Chabini (1998).

A.1 Earliest Origin-to-All TDSP (EOA)

Recall that μ_n represents the EOA label of a node n, the earliest time by which n can be reached $w.p. > 0$. We first define a time-dependent cost parameter $\zeta_{ij}(t) \; \forall \; t \geq t_O, \; i,j \in N$ as follows:

$$\zeta_{ij}(t) = \min \left\{ \min_{\substack{b \in B, \, \beta_b \in I_b, \\ (i,j) \in A(\beta_b): \; t \in T_{\beta_b}(i)}} c^1_{\beta_b}\left((i,j)\right), \; w_{ij} \right\} \tag{11}$$

The cost of arc (i,j) for a departure time t, $\zeta_{ij}(t)$, is set to the minimum of the walking travel time between i and j and the lowest possible cost on the transit arc (i,j) served by buses that reach node i at time t $w.p. > 0$. To illustrate the EOA problem, consider the network shown in Figure 14.

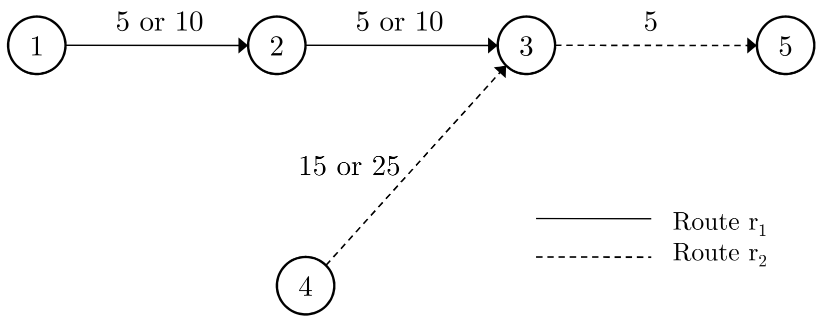


Figure 14: Network to illustrate the preprocessing algorithms

Suppose at $t = 0$, two buses b_1 and b_2 start at nodes 1 and 4 on routes r_1 and r_2 respectively. Let the cost of the transit arcs be as shown in the figure and let the walking travel time along these arcs be 40 (for now, ignore the possibility of walking between other node pairs). Assume a traveler headed to node 5 departs from node 1 (origin) at $t = 0$.

The earliest the traveler can reach the destination is 20, which is possible if b_1 reaches node 3 at $t = 10$ or $t = 15$ and b_2 takes 15 minutes to traverse arc (4,3). Observe that the EOA labels cannot be calculated by assuming that all the buses in the network travel at their fastest pace. For instance, if the cost of 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, if bus b_2 is delayed he/she can reach the destination at $t = 15$.

The pseudocode described below is a label correcting algorithm for one-to-all TDSP with waiting allowed in which the time-dependent arc costs are defined using $\zeta_{ij}(t)$. 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 proof of correctness follows directly from that of an one-to-all TDSP algorithm (see Chabini (1998)).

Algorithm 2 *Pseudocode for EOA*

```
Step 0: Initialize Labels
$\mu_O = t_O$
$\mu_n = \infty \ \forall \ n \in N \setminus \{O\}$
SE = $\{O\}$

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

Assuming that the SE list is implemented using a queue structure, the computational complexity can be shown to be $\mathcal{O}\left(|N|^3|T|\right)$ using the following argument. First, notice that the label of the origin is optimal. When the origin is removed from the SE list, the algorithm scans each of its adjacent nodes for all time periods (which takes $\mathcal{O}\left(|N||T|\right)$ steps) before guaranteeing the optimality of another node label. In this process, the algorithm may add at most $|N|-1$ nodes (the origin never re-enters as the arc costs are non-negative) to the SE list. In order to permanently label a third node, the algorithm must remove each of these $|N|-1$ nodes and scan their adjacency lists which takes $\mathcal{O}\left((|N|-1)|N||T|\right)$. Again, this procedure may add at most $|N|-2$ nodes to the SE list (note that the new nodes are always added at the end of the queue). Proceeding similarly, we may conclude that the algorithm terminates in at most $|N||T|+(|N|-1)|N||T|+(|N|-2)|N||T|+\ldots+(1)|N||T|$ steps, and hence its computational complexity is $\mathcal{O}(|N|^3|T|)$.

A.2 Earliest All-to-Destination TDSP (EAD)

The EAD problem is a straightforward extension of the EOA problem. The time-dependent arc costs $\zeta_{ij}(t)$ defined earlier are used to compute $\gamma_n(t)$, which represents the earliest we can reach the destination *w.p.* > 0, given that we depart from node n at time t. The algorithm for the EAD problem first adds the destination to the SE list and updates the labels of the upstream nodes in order to satisfy the optimality criterion.

When $t \geq T_{max}$, we assume $\gamma_n(t) = \gamma_n(T_{max})$, which reflects a steady state after time T_{max}. This assumption is not restrictive as long as T_{max} is sufficiently large in which case, the time-dependent arc costs equal the walking travel times. Algorithm 3 presents the pseudocode for estimating the EAD labels which is similar to an all-to-one TDSP algorithm with waiting allowed. Note that waiting is modeled using the self-loops of cost 1. The algorithm is guaranteed to converge to the optimal labels and its computational complexity can be shown to be $\mathcal{O}\left(|N|^3|T|^2\right)$ (see Ziliaskopoulos (1994) for a formal proof).

Algorithm 3 *Pseudocode for EAD*

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

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

A.3 Latest Origin-to-all TDSP (LOA)

This problem involves finding λ_n, the earliest time by which we are guaranteed to reach a node n given that we depart from the origin at t_O. To compute these labels, a modified label setting method is applied, which ensures that all transfers are made $w.p.1$ by checking if the *latest arrival time at a transfer or boarding node is less than or equal to the earliest possible departure time of buses serving that node.* Consider the network shown in Figure 14. Clearly, the traveler can reach node 3 at $t = 20$ $w.p.1$. However, the traveler could 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 algorithm on a network with the longest possible transit arc costs. Fixing the travel times on transit arcs to their largest possible values incorrectly updates the label λ_5 to 30 because one can catch the second bus after reaching node 3 at $t = 20$.

Consider the following approach. Given a node i, we check for the optimality of the labels of its downstream nodes (say j), allowing waiting at i, using a time-dependent arc cost set to the minimum of the walking travel time w_{ij} and the largest possible cost on transit arc (i, j) experienced by a bus that can be caught at node i $w.p.1$, assuming that it arrives at its last possible state (i.e., at $l_{\beta_b}(i)$). We begin with the origin and nodes whose labels change can be added to a SE list and this procedure may be repeated till list is empty. However, this approach might fail in some cases as demonstrated in Figure 15.

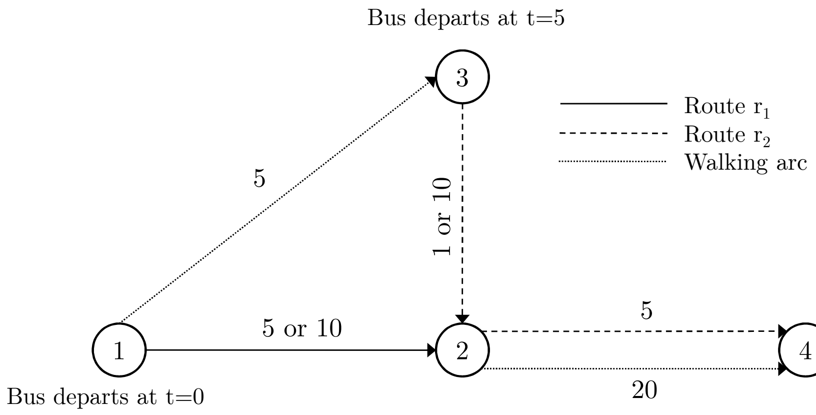


Figure 15: Application of the LOA algorithm

Let two buses b_1 and b_2 start on routes r_1 and r_2 at $t = 0$ and $t = 5$ respectively. Assume the LOA labels are computed for a traveler departing from node 1 at $t = 0$. Following the method described earlier, the labels of nodes 2 and 3 are updated to 10 and 5 respectively. Upon examination of arc (2,4), the algorithm concludes that b_2 cannot be caught with $w.p.1$, and hence the label of 4 is modified to 30 using the walking arc (2,4). Note that one cannot further reduce the label of node 2 using transit arc (3,2), and hence the procedure terminates. The problem with this approach lies in the fact that sub-path optimality does not hold. The traveler can board b_2 $w.p.1$ after walking to node 3, and hence reach the destination at $t = 20$ $w.p.1$. Therefore, when an arc is used to update the label of a downstream node, it is important to check if a bus that traverses the arc could have been boarded at some other node in the network $w.p.1$.

In order to account for such cases, we define variables φ_b, which is set to 1 if a bus b can be boarded $w.p.1$ and φ'_b, which stores the lowest possible *order* of the individual states of b at which it can be boarded $w.p.1$. We initially set φ_b to 0 for all $b \in B$ and when a node is examined, we update φ_b to 1 if a bus traversing one of its downstream arcs can be boarded $w.p.1$. In this way, we can first make sure that φ_b is 1 before using a transit arc served by b to update the label of a downstream node. Let us now apply this method to the network in Figure 15. When node 3 is observed, φ_{b_2} is updated to 1 as we know that b_2 can be surely boarded. Thus, when 2 is examined, the transit arc (2,4) can be used to update the label of node 4. However, the validity of this method hinges on the order in which nodes in the SE list are scanned. Even if they are examined in the increasing order of labels, breaking ties arbitrarily can result in incorrect label updates. Therefore, we scan all the downstream arcs of all the nodes with the minimum label to update φ_b, after which we select a node, remove it from the SE list, and check the optimality conditions for nodes adjacent to it.

We first define a parameter $\varsigma_{ij}^{b}(t)\ \forall\ t \geq t_O, i, j \in N$, which is set to the difference between the latest arrival times at nodes j and i assuming that a bus b serving a particular trip is at its last possible individual state at node i.

$$\varsigma_{ij}^{b}(t) = \begin{cases} l_{\beta_b}(j) - l_{\beta_b}(i) & \text{if } \exists\ \beta_b \in I_b,\ (i,j) \in A(\beta_b) : t = l_{\beta_b}(i) \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

Observe that by Assumption 9, the difference in the latest arrival times in (12) is simply the largest possible travel time experienced by bus b on arc (i, j) during trip β_b.

Let $\Phi_{n,t} \subset B$ be the subset of buses that can be caught *w.p.*1 by a traveler at node n at time t (allowing waiting), that is, $\Phi_{n,t} = \{b \in B \text{ if } \exists\,\beta_b \in I_b : n \in N(\beta_b), t \leq f_{\beta_b}(n)\}$. Also 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 by a traveler at (n, t) *w.p.*1. An indicator variable ϵ_i is used to check if node i was previously examined to update the φ values. Using these parameters and definitions, we propose that the following algorithm computes the optimal LOA labels.

Algorithm 4 *Pseudocode For LOA*

```
**Step 0:** Initialize Labels
$\lambda_O = t_O$
$\lambda_n = \infty\ \forall\ n \in N \setminus \{O\}$
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 $i : \lambda_i = \min_{k \in SE} \lambda_k,\ \ \epsilon_i \neq 1$ do                ▷ Step I
        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 $i : \lambda_i = \min_{k \in SE} \lambda_k$                ▷ Step II
    for each $j \in \Gamma(i)$ do
        for all $t \geq \lambda_i,\ t \in T$ do
            $d_{ij}(t) = w_{ij}$
            for all $b \in B,\ \beta_b \in I_b,\ i \in N(\beta_b) :\ t = l_{\beta_b}(i),\ \varphi'_b \leq order_b\big((i,t)\big)$ 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 I, the algorithm updates the information on buses that can be caught $w.p.1$ from all nodes in the SE list that have the least label and in Step II, it picks a node with the smallest label, scans its adjacency list, and updates their labels (if necessary) using the walking travel time or the largest possible transit travel times of buses that can be boarded $w.p.1$.

Proposition A.1. *Algorithm 4 terminates in a finite number of steps and yields the optimal LOA labels upon termination.*

Proof. Suppose not. If the algorithm fails to terminate, there exists a node that enters the SE list infinitely many times, and hence its label must be unbounded. This leads to a contradiction since the $d_{ij}(t)$'s are non-negative. Hence, the algorithm converges in a finite number of iterations.

Now suppose that the labels obtained from the algorithm are not optimal. Then there exists a node j such that λ_j is not optimal and λ_n is optimal for all $\{n \in N : \lambda_n < \lambda_j\}$. There also exists an optimal path $O - i_1 - \ldots - i_{k-1} - i_k = i - j$ (which consists of waiting, walking, or transit arcs) that guarantees that j can be reached before λ_j *w.p.*1. We allow repetition of nodes when the path contains waiting arcs. Clearly, $\lambda_i < \lambda_j$ and hence λ_i is optimal. But since λ_j is not optimal, it follows that $\lambda_j > \lambda_i + d_{ij}(\lambda_i)$. Note that in the optimal path, the arc from node i to j can either be a walking or transit arc.

If (i, j) represents a walking arc, a contradiction is obtained because the algorithm would have updated λ_j in Step II. Now suppose (i, j) is a transit arc served by bus b. A similar contradiction can be established in this case but we first need to check if the algorithm sets the value of $d_{ij}(\lambda_i)$ to $\varsigma^b_{ij}(\lambda_i)$. Suppose b could have been boarded at a node j' *w.p.*1. Using the fact that $\lambda_{j'}$ is optimal (since $\lambda_{j'} < \lambda_j$), we may conclude that the algorithm would have scanned j' (before node j) and updated φ'_b in Step I. Thus, when node i is scanned, the algorithm would set $d_{ij}(\lambda_i)$ to $\varsigma^b_{ij}(\lambda_i)$ in Step II and the label of j would have been updated. ■

Proposition A.2. *The computational complexity of Algorithm 4 is* $\mathcal{O}\left(|N|^2|T||B|\right)$.

Proof. When the algorithm first scans the origin's adjacency list, at most $|N| - 1$ nodes may be added to the SE list. In order to permanently label the next node, the algorithm uses Steps I and II which require $\mathcal{O}(|N||B|)$ and $\mathcal{O}(|N||T||B|)$ computations respectively. Since nodes do not re-enter the SE list, permanently labeling the third node requires $\mathcal{O}((|N| - 1)|B| + |N||T||B|)$ steps. Repeating this argument, we may deduce that $|N||B| + (|N| - 1)|B| + \ldots + (1)|B| + |N|\left(|N||T||B|\right)$ is an upper bound on the total number of computations and therefore, the complexity of Algorithm 4 is $\mathcal{O}\left(|N|^2|T||B|\right)$. ■

Alternately, the LOA labels can be computed using a recursive application of a TDSP algorithm by varying the time-dependent arc costs. We first set the time-dependent arc costs to the 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 *w.p.*1. The time-dependent arc costs are then updated assuming that the bus takes the longest possible time to traverse arcs on trips beyond the stop at which it could be boarded *w.p.*1. The TDSP labels are computed again and this process is repeated until the labels do not change.

Notice that if Assumption 9 is relaxed, the differences between the latest arrival times in (12) need not be equal to the largest possible transit travel times. However, in such cases, using the definition of $\varsigma_{ij}^{b}(t)$, the reader may easily verify that the optimal LOA labels can still be obtained using Algorithm 4.

A.4 Latest All-to-Destination TDSP (LAD)

The LAD problem for each individual state of a bus finds the earliest we can reach the destination *w.p.*1, assuming that the state of a traveler is same as that of the bus. To compute these labels (which are denoted by $\eta(s_b)$), we make minor modifications to the input parameters of the LOA algorithm and solve it repeatedly for each individual state. Since the running time of the LOA algorithm is $\mathcal{O}\left(|N|^2|T||B|\right)$, the complexity of the LAD problem is $\mathcal{O}\left(\left(\sum_{b\in B}|S_b|\right)|N|^2|T||B|\right)$.

Let us revisit the example in Figure 14. The individual states and their corresponding LAD labels are shown in Figure 16 for both buses b_1 and b_2. Consider the state (2,5) for the bus on the first route, i.e., b_1 is at node 2 at $t = 5$. As the traveler can board b_1, he/she can reach node 3 at $t = 15$ *w.p.*1, and can successfully transfer to b_2 and reach the destination at $t = 30$ *w.p.*1.

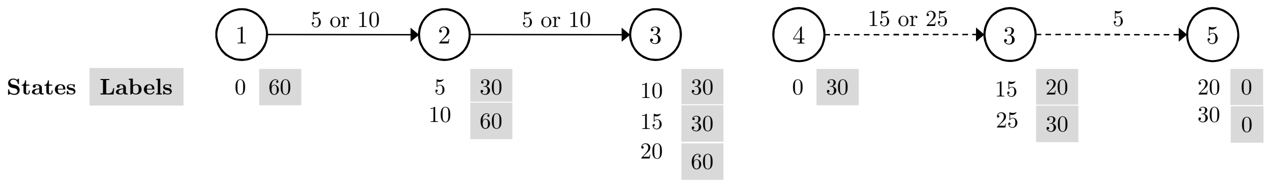


Figure 16: LAD labels of individual states of bus b_1 (left) and b_2 (right)

Since the traveler's individual state is same as that of the bus b, he/she can board b *w.p.*1. Therefore, in the initialization step of the Algorithm 4, we set $\varphi_b = 1$ and $\varphi'_b = order_b(s_b)$. Note that the origin of the traveler is $\tilde{n}(s_b)$ and hence $\lambda_{\tilde{n}(s_b)} = \tilde{t}(s_b)$. The LOA algorithm may then be used to calculate the label of the destination (which is the LAD label for state s_b).

List of Figures

References

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

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

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

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

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

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

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

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

Huang, H. and Gao, S. (2012). Optimal paths in dynamic networks with dependent random link travel times. *Transportation Research Part B: Methodological*, 46(5):579 – 598.

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

Miller-Hooks, E. D. and Mahmassani, H. S. (1998). Optimal Routing of hazardous materials in stochastic, time-varying transportation networks. *Transportation Research Record*, 1645:143–151.

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

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

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

Nguyen, S., Pallottino, S., and Gendreau, M. (1998). Implicit enumeration of hyperpaths in a logit model for transit networks. *Transportation Science*, 32(1):54–64.

Nielsen, L., Pretolani, D., and Andersen, K. (2004). K shortest paths in stochastic time-dependent networks. Technical Report WP-L-2004-05, Department of Accounting, Finance and Logistics, Aarhus School of Business.

Nielsen, L. R., Andersen, K. A., and Pretolani, D. (2003). Bicriterion shortest hyperpaths in random time-dependent networks. *IMA Journal of Management Mathematics*, 14(3):271–303.

Nielsen, L. R., Andersen, K. A., and Pretolani, D. (2014). Ranking paths in stochastic time-dependent networks. *European Journal of Operational Research*, 236(3):903 – 914.

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

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

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

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

Rambha, T. (2012). Adaptive routing in schedule based stochastic time-dependent transit networks. Master's thesis, The University of Texas at Austin.

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

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

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

Ziliaskopoulos, A. K. (1994). *Optimum Path Algorithms on Multidimensional Networks: Analysis, Design, Implementation and Computational Experience*. PhD thesis, The University of Texas at Austin.