EFFICIENT OPERATIONAL STRATEGIES FOR RIDESHARING SYSTEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yang Liu

May 2019

EFFICIENT OPERATIONAL STRATEGIES FOR RIDESHARING SYSTEMS

Yang Liu, Ph.D.

Cornell University 2019

This dissertation aims at developing computationally tractable models and algorithms to increase the operational efficiency of the Mobility-on-Demand (MoD) ridesharing systems via three means: i) fast ride-vehicle matching; ii) proactive rebalancing; iii) request-level pricing strategies, and iv) framework to incorporate mode choice models.

Mobility-on-Demand (MoD) services such as Uber and Lyft are disrupting urban mobility. Over the last few years, this disruption has led to much interest in studying the design, management and impacts of these systems. However, due to the large scale of the demand and fleet size, how to optimally assign the vehicles to trip requests and how to optimize the prices charged on riders in real time still remain as open problems. Studies usually solve the request-vehicle assignment task in two ways: i) greedy matching, in which a new coming request is matched to a request *immediately*; ii) batching matching, in which the requests are collected for a short time window (e.g. 30 seconds), and at the end of the time window the optimization problem is considered for the whole batch of requests. Batching matching can usually increase the matching efficiency since it can use the fleet better compared to the greedy matching. However, riders' waiting time can be prolonged due to the batching window, and damage the riders' experience. There is not a consensus on which method is better, and this dissertation discusses both frameworks and one can implement the methods and compare the performance via simulation. In addition, the research proposes speed-up

techniques for the batching matching, which can reduce the computation time and increase the matching efficiency given a limited time window.

MoD systems are generally designed and analyzed for a fixed and exogenous demand, but such frameworks fail to answer questions about the impact of these services on the urban transportation system, such as the effect of induced demand and the implications for transit ridership. In this dissertation, we propose a unified framework to design, optimize and analyze MoD operations within a multimodal transportation system where the demand for a travel mode is a function of its level of service. An application of Bayesian optimization (BO) to derive the optimal supply-side MoD parameters (e.g., fleet size) is also illustrated.

The profit of the MoD companies and the balance between the supply and demand highly depend on the pricing strategies applied. Since the supply and demand volumes at different locations change continuously, dynamic pricing has been proven to be more robust and be helpful to relieve the Wild Goose Chase (WGC) phenomenon. However, the current dynamic pricing method is usually state-dependent. Specifically, given a location zone and its supply level at the moment, the system determines the price multiplier for this zone for the next time window. Although such methods are easy to implement, they ignore the network effects among different location zones, and also they cannot differ the price (multiplier) for requests in the same zone. In practice, travelers make mode choice decisions by comparing the characteristics of each service in the market, and the characteristics of the alternatives (e.g. public transit service) are usually different for each traveler due to their different destination and sociodemographic characteristics. To overcome the above difficulty, the pricing method presented in this dissertation determines the price for each trip request

individually.

Although MoD services have become one of the major transportation modes, public transit service is still an affordable and clean transportation mode choice. This dissertation also develops fast computation techniques to solve the reliable routing problem known as the stochastic on-time arrival (SOTA) problem in transit networks, which provides a routing strategy that maximizes the probability of arriving at the destination within a given time budget.

BIOGRAPHICAL SKETCH

Yang Liu did his undergraduate studies in Management Information Systems at Tianjin University. After that, he continued his study at the university and got a master degree in Management Science and Engineering in 2012. At 2015, he continued his research career and began to pursue a Ph.D. degree in Transportation Systems Engineering at Cornell University.

To my parents who provided me with the opportunity to have the best education.

ACKNOWLEDGEMENTS

The research presented in this dissertation would not have been possible without the help and support of many people in the last four years. I would like to take this opportunity to express my deepest appreciation to all those involved.

I would like to begin by thanking my advisor, Professor Samitha Samaranayake, for providing me with the opportunity to work in his research group and all the support he gave me during my time at Cornell. He continuously and convincingly conveyed the spirit of adventure regarding research, and his patient guidance, encouragement and advice have affected me through my time as his advisee. He helped me build research collaborations, and kept demonstrating what good research and good researchers look like. Without his persistent help, this dissertation would not have been possible.

I would also like to thank my special committee members, Professor Oliver Gao and Professor Siddhartha Banerjee. I was fortunate to work with and have the great advice from those brilliant people. Their deep knowledge and vision greatly helped me find the method to solve the problems in this dissertation.

In addition, I was lucky to collaborate with many excellent scholars here, and I would like to take this opportunity to express my deepest gratitude. First, I want to thank Dr. Sebastien Blandin, who gave me a lot of detailed advice in my first research project and let me have a good start at Cornell. Then, I would like to thank Prateek Bansal, who kept affecting me with his boundless energy and his openness to share his insights. I truly believe that he has all the abilities to become a rising star in future academia. Finally, I thank Xiaotong Guo and Dr. Raga Gopalakrishnan, who provides me with the opportunity to work with them on those wonderful research projects.

During my Ph.D. time, I was also lucky to have the opportunity to be an

intern at Facebook, inc. I would like to thank my mentor Bobby Fortanely and my manager Guy Lavie. They not only demonstrated the standard of being a good engineer, but also showed their true care for me. I will never forget all the support and encouragement from them in my future career. I also want to acknowledge all the colleagues and interns that I had the pleasure to collaborate with at Facebook.

Thanks to Dr. Chao Wang, Xiao Xu and Boshuang Wang and all the friends for all the good times we shared at Cornell. I would not be able to complete the journey without all those companions. I would also like to thank Professor Ning Zhu and Professor Shoufeng Ma, who were my advisors when I was a master student. It was them who guided me to start this path, and I am deeply appreciative of all those guidances and the subsequent support.

Finally, I would like to thank my parents. I am fortunate to have all the love, support and I would not been here without your countless sacrifices through the years.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The popularity of ridesharing platforms such as Lyft and Uber has grown them into multi-million dollar markets [4]. These Mobility-on-Demand (MoD) services not only provide additional travel options to passengers but also have the potential to decrease auto ownership and pollution. High capacity ridesharing services such as Via also offer affordable alternatives to public transit services.

The efficiency of the MoD systems highly depends on the operational strategies of the company, which consists of strategies in different levels: to maximize the profit, the company needs to: i) in the system level, the company needs to determine the optimal fleet size of each service it provides; ii) in the network level, the company has to assign the vehicles to each request, but also needs to maintain the balance of the supply and the demand, and iii) at request level, the company needs to determine the price for each trip. All these decisions affect the profit of the company and the welfare of travelers.

Compared to the traditional taxi service, the MoD dispatch system can match the vehicles to trip requests in an online manner without a human dispatcher, which makes it possible to optimize the dispatch at scale [118]. However, what is the best method to match the vehicles to requests still remain as an open question, and for high capacity MoD services, it is difficult to find the optimal matching in real time due to the large scale of the demand and fleet size.

When a passenger plans a trip using a ridesharing mobile application, the trip price and some other information about the trip are displayed to the travelers. The passenger has the freedom to cancel the trip at no cost. The decision of the passenger to accept or cancel the trip generally depends on the displayed wait time and their perception of trip cost. Therefore, good pricing strategies can not only increase the profit of the company, but also help attain the balance of the supply and the demand.

The research presented in this dissertation is motivated by the need for computationally tractable and efficient operational strategies for MoD services. Therefore, models about MoD services are tested using real taxi demand data in Manhattan, New York City. In the experiments for the SOTA problem, the model is tested using real transit network in Chicago.

1.2 Fleet management for high capacity Mobility-on-Demand (MoD) services

The fleet management strategies consist of the assignment from the vehicles to the requests and the rebalance method for the idling vehicles. MoD services are currently operated by vehicles with human drivers. However, the fleet will probably to comprise autonomous vehicles since millions of dollars are invested in the autonomous vehicle industry these years. Google planned to build a fleet of autonomous vehicles [74], and in Europe, low-speed (25 miles per hour) twelve-seat automated vehicles deployments were also underway through the CityMobil2 project [30]. With such advancements in automation technologies, demand-responsive autonomous mobility services are also gaining research in-

terest [22, 30, 58, 102].

To provide high-quality on-demand ridesharing service, efficient fleet management methods to optimize the assignment from the vehicles to travel requests are required. Most of the literature on Mobility-on-Demand (MoD) systems have been focused on the case without pooling travel requests until recently [103, 120]. A recent study showed that 80% taxi trips in Manhattan, New York City (NYC) could be shared by two riders with only a few minutes travel time increase [97], which quantified the benefits of vehicle pooling on a real-size network. However, the method only provides the optimal solution when a ride can be shared by at most 2 riders, and is intractable for high capacity vehicle pooling. To overcome this difficulty, a computationally efficient anytime optimal fleet management algorithm was proposed to address both the problem of assigning vehicles to travel requests and the problem of rebalancing the idling vehicles for a high capacity fleet [1]. Traditional methods for fleet management problem usually formulate the problem as an Integer Linear Program (ILP), which is not tractable for large-scale instances due to the large solution space. The method in [1] decoupled the problem by first employing the pairwise shareability graph [97] to compute feasible trips and then using a compacter ILP to find the optimal assignment from the vehicles to feasible trips. Extensive experiments using taxi trips in Manhattan, NYC showed that the algorithm could provide a satisfying solution in real time for most of the cases. However, the computation time could still be high when the fleet size was large. For example, considering a fleet of 3000 four-seat vehicles, the simulation for a day's travel demand could be as high as about 41 hours using a 24-core 2.5GHz computer [2]. In addition, MoD system performance can benefit from a more efficient vehicle rebalancing method since the method in [1] was simply match-

ing idle vehicles to the locations of the unserved requests assuming that these requests may request again.

The first part of this dissertation (Chapter 2) is to develop techniques to improve the state-of-the-art framework in [1], which includes: i) search space pruning techniques when computing feasible trips; ii) a parallelization Input/Output (I/O) reduction technique; iii) improved vehicle assignment and vehicle rebalancing formulation; iv) a proactive rebalancing method. These techniques would not only help researchers conduct simulation-based studies more quickly but also provide insights for the implementation of the method in the industry.

1.3 Optimal design of Mobility-on-Demand (MoD) services at system level

The optimal design of MoD systems at the system level will not only affect the profit obtained by the company but also the welfare of the travelers in the system. However, most existing models assume a fixed and exogenous demand for MoD services by considering a waiting time threshold. In practice, this may not be true: MoD services coexist with other travel modes (e.g., transit and walking). The demand for MoD is not only a function of its attributes (e.g., price, travel time), but also depends on characteristics of the competing travel modes as well based on individual preferences.

In Chapter 3, we develop a unified framework which integrates mode choice models with a state-of-the-art system for modeling real-time on-demand mobil-

ity services with varying passenger capacities (based on [1]). In addition, we implement and integrate a Bayesian Optimization (BO) based solver to find the optimal supply-side MoD parameters (e.g., fleet size, fare, etc).

In summary, the proposed framework would not only help in better understanding the influence of MoD services on urban mobility and answer system-level policy questions, but would also illustrate a method to tune the supply-side parameters that influence the demand.

1.4 Optimal pricing for Mobility-on-Demand (MoD) services at trip request level

Pricing is one of the most important tools to affect the interaction between the supply-side and the demand side in ridesharing systems, which has received considerable attention recently. Most of the existing pricing frameworks are state-dependent, which aim to find the optimal pricing multiplier for a location zone given the supply level for a time window [15, 55]. Such dynamic pricing methods are easy to implement, but they ignore the network effects and cannot impose different prices for requests in the same zone. In addition, it is not clear how to use such models to find pricing strategies for multiple services due to the substitution behavior across MoD products.

In this dissertation, we consider a multimodal transportation system, where three travel modes coexist – the exclusive MoD service (e.g., UberX), the sharing MoD service (e.g., Uberpool), and the taxi service. The above two MoD services are operated by one service operator. The objective is to optimize the prices of

both services for each trip request and the vehicle assignment to maximize the operator's profit. We assume that demand is endogenous in the system, and the travelers will make mode choice according to an estimated multinomial logit model [64, 90]. We develop: i) A sequential MoD service pricing framework, which can be integrated in the current ridesharing service (Uber, Lyft, etc) usage scenario, and ii) A batching MoD service pricing framework, which optimizes the prices of MoD services and vehicle assignment for the trip requests in each time interval. In both frameworks, we connect the pricing problem for each request to the profit of the future time window, which makes the strategies less myopic.

1.5 Stochastic on-time arrival (SOTA) problem in transit networks

People navigate over 1 billion kilometers a day using mobile routing services [109], and many of these services provide travelers with information on transit networks. In most of such applications, given an origin-destination (OD) pair and a desired departure or arrival time, the route associated with the minimum expected trip time is provided to the user. However, the uncertainty associated with these recommendations, either due to variability in travel time or the transit service headway, is rarely accounted for. In this dissertation, we also attempt to bridge this gap by formulating and solving the Stochastic On-time Arrival (SOTA) problem for transit networks, which provides a transit routing policy that maximizes the probability of reaching the destination within a given travel time budget in stochastic transit networks.

Given the complexity of the SOTA problem, significant efforts have been made to design efficient solution algorithms in road networks. In transit networks, the problem of determining a set of reliable travel decisions is significantly more complex than in road networks. In this context, the traveler is not in control of the transit line(s) that they may travel on to reach the destination, and may have to make a number of complex decisions regarding which transit line(s) to take. In our work, we consider the routing problem in transit networks as a fully online problem. The online information setting mentioned above can be easily adopted into our framework by changing the travel time and headway distributions accordingly. Specifically, we propose the formulation of the SOTA problem for transit networks, including a general network structure for stochastic transit networks and a decision-making model. Both the waiting time for each transit line and the travel time on each link are assumed to be random variables with known probability density functions. To find the optimal routing policy, we design a dynamic programming (DP) algorithm, which is pseudo-polynomial in the number of transit stations and time budget, and exponential in the number of transit lines at each station. To reduce the search space, we develop a definition of transit line dominance and present methods to identify this transit line dominance, which significantly decreases the computation time in our numerical experiments.

CHAPTER 2

FLEET MANAGEMENT FOR HIGH CAPACITY MOBILITY-ON-DEMAND (MOD) SERVICES

2.1 Fleet management problem for high capacity MoD services

2.1.1 Introduction

The research presented in this chapter aims to improve the state-of-the-art fleet management approach in [1] in both computation performance and system performance. All the techniques can be integrated in the framework without affecting any time optimality.

This chapter first presents a number of numerical results for improving the computational tractability of the problem. We first propose two search space pruning techniques: the first technique significantly reduce duplicate computation by employing the fact that the travel time on the link will not change dramatically and the second technique is an early stopping criterion when checking the feasibility of a trip. Since most of the computation in the framework is paralleled on different process workers, we also reduce the input/output overhead using K-means clustering method. In addition, the formulation of the vehicle assignment and the rebalancing are also improved by reducing some of the constraints without affecting the quality of the solution

The rebalancing method used in the framework in [1] guides the idle vehicles to the location where requests are ignored in the past time window. In this dissertation, we propose a proactive rebalancing algorithm, which can guide

idle vehicles to future demand based on a probability distribution estimated using historical data.

The rest of this chapter is structured as follows. In Section 2.1.2 and Section 2.1.3, we briefly review the fleet management method in [1]. In Section 2.1.4, we introduce speed-up techniques to improve computation performance. Section 2.2 proposes a proactive rebalancing method to improve MoD system performance. Section 2.3 consists of numerical experiments on the Manhattan network to show the efficiency of the techniques.

2.1.2 Problem formulation

We consider a fleet of vehicles with varying passenger capacities to satisfy the demand for MoD services. The operation of MoD services includes two main tasks: i) match travel requests to vehicles; ii) rebalance the idle vehicles to areas with potential future demand. Our formulation follows a state of the art framework which was recently proposed by [1].

The set of vehicles is denoted by $\mathcal{V} = \{v_1, ..., v_n\}$. Each vehicle v_i has a corresponding capacity c_i. The set of trip requests is denoted by $\mathcal{R} = \{r_1, ..., r_n\}$. Multiple travelers are allowed to share one single ride. A *passenger* is defined as a past request that has been picked up by a vehicle and is now en-route to its destination. For each request r, the waiting time is w_r, which is the difference between the pick-up time t_r^p and the request time t_r^r. For each picked up request r, the total travel delay is defined as $\delta_r = t_r^d - t_r^*$, where t_r^d is the drop-off time and t_r^* is the earliest possible time at which the destination could be reached. t_r^* is computed by following the shortest path between the origin o_r and the desti-

nation d_r with zero waiting time. We want to find the optimal assignment that minimizes a cost function C under a set of constraints as follows:

- For each request r, the waiting time w_r must be below the maximum waiting time Ω_r. If one customer has waited for a time more than Ω_r, the request is discarded in the simulation, and a large constant penalty will occur in the cost function.
- For each picked up request r, the total delay δ_r must be lower than the maximum travel delay Δ_r.
- For each vehicle v at any time, the number of passengers in the vehicle must not be larger than the capacity c_v
- The requests for travel via a specific MoD service type (e.g. ride-hailing service, ridepooling service or micro-transit service) can only be served by the vehicle fleet corresponding to the same service type.

The cost function C is set to the sum of total delay and the penalty for unassigned requests in this work[1]. The total delay includes both the waiting time for all assigned requests and the in-vehicle delay caused by sharing with other passengers. If a request is not served under the above constraints, it is discarded in the simulation, and a large penalty c_{ko} will occur in the cost function.

$$C = \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{P}_v} \delta_r + \sum_{r \in \mathcal{R}_{ok}} \delta_r + \sum_{r \in \mathcal{R}_{ko}} c_{ko} \tag{2.1}$$

[1]We use this cost function to be consistent with the framework in [1]. In practice, we can use other cost functions. For example, ridesharing service companies may want to minimize the expected profit.

where $\mathcal{P}_v$ is the set of passengers in vehicle v, $\mathcal{R}_{ok}$ is the set of requests that have been assigned vehicles, and $\mathcal{R}_{ko}$ is the set of requests that are not assigned any vehicles.

2.1.3 State-of-the-art fleet management approach

The MoD operation problem is essentially the same as the dynamic pickup and delivery problem [98] with time windows, which has shown to be difficult to solve exactly for a large-scale demand [81, 93]. The anytime optimal algorithm in [1] decouples the problem by first computing feasible trips from a pairwise shareability graph [97] and then finding the optimal trip assignment to vehicles by solving an Integer Linear Program (ILP) of reduced dimensionality. In this chapter, the assignment of the trip requests is processed at a frequency of every 30 seconds. Requests that have not been picked up in an assignment round remain in the request pool until the maximum waiting time constraint is violated.

Specifically, each round of the assignment simulation includes the following steps:

(1) Compute a pair-wise request-vehicle graph (RV-graph). RV-graph describes possible pairwise matching between vehicles and pick-up requests. In the graph, we connect two requests r_i and r_j if an empty virtual vehicle at the origin of one of them can serve both of them without violating any of the constraints; Similarly, we connect a request r to a vehicle v if v can serve r under the constraints listed in Section 2.1.2.

(2) Construct the Request-Trip-Vehicle graph (RTV-graph) using the cliques

of the RV-graph, which includes all the feasible trips and also the vehicles that can serve them. A feasible trip is defined as a set of requests that can be served by one vehicle under the constraints. In the RTV-graph, a request r is connected to a trip T if T contains r; A trip T is connected to a vehicle v only if v can complete T without violating any constraints. The feasible trips and the edges in the graph are computed incrementally in trip length for each vehicle, which successfully reduces the search space [1]. This step results in all the feasible assignments between the vehicles and the requests.

(3) Solve an ILP to compute the optimal assignment of vehicles to trips. Since each trip can possibly be served by multiple vehicles, the ILP needs to not only determine the assignment, but also ensure that each request and vehicle are assigned to only one trip. The ILP in [1] is as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i,j \in X_{TV}} c_{i,j} \cdot x_{i,j} + \sum_{k \in \{1,\dots,n\}} c_{ko} \cdot \chi_k \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}^T_{V=j}} x_{i,j} \leq 1, && \forall v_j \in \mathcal{V} \\
& \sum_{i \in \mathcal{I}^T_{R=k}} \sum_{j \in \mathcal{I}^V_{T=i}} x_{i,j} + \chi_k = 1, && \forall r_k \in \mathcal{R} \\
& x_{i,j} \in \{0,1\}, && \forall i,j \in X_{TV}
\end{aligned} \tag{2.2}$$

where X_{TV} is the set of all feasible assignments between trips and vehicles, $c_{i,j}$ is the cost of vehicle j serving trip i, c_{ko} is the constant penalty for an unassigned request, which controls the trade-off between the total delay term and the penalty term. In our experiments, c_{ko} is set to a large constant to make sure that the objective is to minimize the total delay given that the maximum possible number of requests are served. If c_ko is set to a relatively small value, some requests may be rejected since the cost of serving them may be higher than the

penalty for not serving them. The decision variables are $x_{i,j}$ and χ_k, where $x_{i,j} = 1$ if vehicle j is assigned to trip i, and $\chi_k = 1$ if the request r_k is unassigned in this round of assignment simulation. In the constraints, there are three sets: the set of trips that can be served by vehicle j is $\mathcal{I}^T_{V=j}$; the set of trips that contains request r_k is $\mathcal{I}^T_{R=k}$; the set of vehicles that can serve trip i is $\mathcal{I}^V_{T=i}$. The constraints ensure that: i) each vehicle will serve at most one trip; ii) each request is either served by one vehicle or ignored ($\chi_k = 1$) in the assignment.

(4) Rebalance the idle vehicles. After the assignment, there may be idle vehicles (no trips assigned to the vehicle) and unsatisfied requests (no vehicles assigned). Assuming that more requests will be likely to appear in the neighborhood of the unsatisfied requests, a linear program (LP) is solved to match the idle vehicles to the locations of unsatisfied requests while minimizing the rebalancing cost.

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in \mathcal{V}_{\text{idle}}} \sum_{r \in \mathcal{R}_{ko}} \tau_{v,r} y_{v,r} \\ \text{subject to} \quad & \sum_{v \in \mathcal{V}_{\text{idle}}} \sum_{r \in \mathcal{R}_{ko}} y_{v,r} = \min(|\mathcal{V}_{\text{idle}}|, |\mathcal{R}_{ko}|) \\ & 0 \leq y_{v,r} \leq 1 \quad \forall y_{v,r} \in \mathcal{Y} \end{aligned} \tag{2.3}$$

where $\mathcal{V}_{\text{idle}}$ is the set of idle vehicles, $\tau_{v,r}$ is the travel time between the vehicle v's location and the origin of the unassigned request r, $y_{v,r}$ is the decision variable where $y_{v,r} = 1$ if the vehicle v is assigned the task to travel to the origin of r and 0 otherwise, and $\mathcal{Y}$ is the set of decision variables.

2.1.4 Speed up techniques

In this section, we propose speed-up techniques to improve the on-demand high capacity vehicle pooling framework. The techniques in this section do not impair the anytime optimality of the framework. Readers can also refer to [27] for approximation methods.

Search space pruning

In [1], an exhaustive search[2] is conducted to check if a vehicle can serve a trip. It is time-consuming for a trip with a high number of requests. In addition, the number of trips increases exponentially with the number of requests. Therefore, we propose techniques to: i) reduce the number of times required to conduct an exhaustive search; ii) decrease the computation time of each exhaustive search.

Claim 1. *Given a network with static travel time, if no vehicle can serve trip T under the constraints in the current round of assignment simulation, there will not be any vehicles that can serve trip T in the future.*

As the travel time is not changing dynamically, the above claim is intuitively true. In [1], when a trip is not feasible for vehicle v in the current round of assignment simulation, it is still considered for v in the following rounds. To reduce such duplicate computation, we propose the following technique:

Claim 2. *For any trip i, an array A_i can be used to record the set of feasible vehicles. In the following rounds of assignment simulation, only the vehicles in set A_i are considered for trip i. We keep updating A_i, and discard trip i whenever $|A_i| = 0$.*

[2] Enumerate every possible order of pickups and dropoffs for the requests to find the optimal route and check if it violates any constraints.

As the number of trips increases exponentially with the number of requests, it can be memory-consuming if we consider all trips. In addition, the set of feasible vehicles is small for a trip of multiple requests. Therefore, the computation time saved by the technique is diminishing as the length of the trip increases. In our work, we use the technique only for the trips with one request. Given a network with dynamic travel time, the technique can be used as an efficient approximation since the travel time will not change significantly during a trip.

The exhaustive search in [1] checks the feasibility of every possible route (order of pickups and dropoffs for the requests in the trip) for a trip. For each route, the framework lets the vehicle virtually follow it and check the constraints during the process. In [1], the waiting time constraint is checked when traversing a request's origin, and the delay constraint is checked when traversing a request's destination. The following technique can potentially decrease the computation time of the exhaustive search process for each route.

Claim 3. *During checking a trip's feasibility, whenever the vehicle traverses an origin or a destination, the constraints are checked for every node in the route that has not been traversed. If any constraint is violated, the route is not feasible.*

The intuition is that when the vehicle arrives at some new location, some requests may already be infeasible due to the vehicle's detour on the prior route section, and continuing to follow the route causes redundant computation.

Parallelization I/O reduction

In [1], when constructing the RV-graph, each request is connected to all the vehicles that can serve it under the constraints. Specifically, for each request-vehicle

pair, the framework checks if exists a feasible route to serve the request and all the passengers in the vehicle. This is conducted via the exhaustive search and is parallelized among the requests. Each request is assigned to a worker in the pool of worker processes, and the set of possible vehicles is also input to it. Here *possible vehicles* represents that if the vehicle travels directly to the origin of the request, it can pick up the request within the waiting time constraint. As the set of possible vehicles for different requests may overlap with each other, each worker can reduce I/O communication by processing multiple requests in a batch. To minimize the I/O overhead, we propose the problem of optimizing the request-worker assignment:

$$\text{minimize} \quad \sum_{i \in \mathcal{N}} |\bigcup_{r|z_{i,r}=1} \mathcal{V}_r| \tag{2.4}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{N}} z_{i,r} = 1 \qquad \forall r \in \mathcal{R} \tag{2.5}$$

$$z_{i,r} \in \{0, 1\} \qquad \forall i \in N, r \in \mathcal{R} \tag{2.6}$$

where $\mathcal{N}$ is the set of workers, $\mathcal{V}_r$ is the set of possible vehicles for request r, $z_{i,r}$ is the decision variable, where $z_{i,r} = 1$ represents that request r is assigned to worker i, and 0 otherwise. The aim is to minimize the number of vehicles input to the workers in total. The problem is similar to the weighted set partitioning problem but more difficult since the weight for each set is not known before $z_{i,r}$ is set.

The set of possible vehicles should be similar for two requests that are spatially close to each other, Therefore, instead of solving the above problem di-

rectly, we use the K-means clustering algorithm [71] to cluster the requests based on their coordinates, and assign each cluster of requests to different workers. The number of clusters k is chosen according to the number of workers in the computer.

Improved ride-vehicle assignment formulation

In this work, we use the following formulation for trip-vehicle assignment which is mentioned in [65]. The formulation reduces the number of variables by $|\mathcal{R}|$ compared to the formulation discussed in Section 2.1.3.

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j \in X_{TV}} (c_{i,j} - c_{ko} \cdot l_i) \cdot x_{i,j} \\ \text{subject to} \quad & \sum_{i \in \mathcal{I}^T_{V=j}} x_{i,j} \leq 1, && \forall v_j \in \mathcal{V} \\ & \sum_{i \in \mathcal{I}^T_{R=k}} \sum_{j \in \mathcal{I}^V_{T=i}} x_{i,j} \leq 1 && \forall r_k \in \mathcal{R} \\ & x_{i,j} \in \{0,1\}, && \forall i,j \in X_{TV} \end{aligned} \tag{2.7}$$

where l_i is the number of requests in trip i. We track l_i for each trip i. We consider all the requests unassigned before the assignment, and thus a total penalty $c_{ko} \cdot |\mathcal{R}|$[3] is imposed to the objective function. Whenever we assign a vehicle j to a trip i, the system gets a reward $c_{ko} \cdot l_i$ for picking up l_i requests in the trip. The number of variables is reduced by $|\mathcal{R}|$ because we do not need χ_k for each request in the formulation, but the solution is equivalent.

[3]This term is ignored in the objective function since adding constants does not affect the optimal solution.

2.2 Proactive rebalancing for ridesharing fleets

In this section, we propose a proactive rebalancing vehicle rebalancing framework, which can potentially improve MoD system performance. The framework comprises the following changes: i) we use a new rebalancing formulation, which guarantees that the solution is a matching between the idle vehicles and unsatisfied requests; ii) we propose a way to incorporate proactive rebalancing in the rebalancing formulation.

New vehicle rebalancing formulation

The rebalancing algorithm in [1] is conducted in each round of assignment simulation. A vehicle may be assigned different rebalancing tasks in consecutive rounds, and circle around inefficiently in the network. Therefore, we first add a constraint to ensure that if a vehicle is assigned a rebalancing task in previous rounds of simulation, it will not be considered in the rebalancing procedure in the current round of simulation.

In the original formulation discussed in Section 2.1.3, multiple vehicles can be assigned to one unsatisfied request if the request is close to these vehicles. This can make the vehicles rebalance to one neighborhood and decrease the spatial coverage of the fleet. We add one constraint to avoid this disadvantage.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{v \in \mathcal{V}_{\text{idle}}} \sum_{r \in \mathcal{R}_{ko}} \tau_{v,r} y_{v,r} \\
\text{subject to} \quad & \sum_{v \in \mathcal{V}_{\text{idle}}} \sum_{r \in \mathcal{R}_{ko}} y_{v,r} = \min(|\mathcal{V}_{\text{idle}}|, |\mathcal{R}_{ko}|) \\
& \sum_{v \in \mathcal{V}_{\text{idle}}} y_{v,r} \leq 1, \quad \forall r \in \mathcal{R}_{ko} \\
& 0 \leq y_{v,r} \leq 1, \quad \forall y_{v,r} \in \mathcal{Y}
\end{aligned}
\tag{2.8}
$$

Note that in this section we are still rebalancing idle vehicles to the locations of unassigned requests as in [1]. We will replace R_{ko} by potential future requests in Section 2.2. The new formulation ensures that the solution is a valid matching between the vehicles and the requests. However, it increases the number of constraints by $|\mathcal{R}_{ko}|$, and for each of these constraints, we have to calculate a summation of $|\mathcal{V}_{\text{idle}}|$ terms. Formulating the problem for the solver will be time-consuming when $|\mathcal{R}_{ko}|$ is large. In our experiments, we set an upper bound to both the number of vehicles and requests in the formulation, which are denoted by $V^{\max}$ and $R^{\max}$ respectively. Specifically, this is done via the following steps:

Step 1: If $|\mathcal{R}_{ko}| > R^{\max}$, we randomly sample $R^{\max}$ requests from $\mathcal{R}_{ko}$ as the set of requests using in the rebalacing procedure;

Step 2: If $|\mathcal{V}_{\text{idle}}| > \min(V^{\max}, \gamma \cdot \min(|\mathcal{R}_{ko}|, R^{\max}))$, we randomly sample $\min(V^{\max}, \gamma \cdot \min(|\mathcal{R}_{ko}|, R^{\max}))$ vehicles from $\mathcal{V}_{\text{idle}}$ as the set of vehicles using in the rebalance.

We apply a constant multiplier γ, which represents that at most γ idle vehicles are considered on average for each unassigned request. The parameters $R^{\max}$, $V^{\max}$ and γ control the tradeoff between the computation time and the rebalancing optimality at the moment.

Proactive rebalancing

The rebalancing algorithm in [1] assumes new requests will appear in the same areas where the unsatisfied requests are. However, the assumption may not be true if the demand does not show short-term recurring patterns. Therefore, we propose a way to conduct proactive rebalancing based on the probability distribution estimated using historical data.

A recent study proposes a predictive routing method for MoD system by adding virtual future requests according to a prediction method [3]. The predictive routing method builds a probability distribution over the future demand according to historical data. Then, it samples virtual requests based on the distribution, and add them to the request pool. These virtual future requests will guide the vehicles to these areas in the assignment simulation. However, we note from the results that: i) the service rate (number of requests serviced) is approximately the same as the reactive method without virtual request samples. A potential reason is that after sampling, the probability for virtual requests to appear in reality is not considered. Some virtual requests may be rare historically, which harm the system performance; ii) the computation time increases significantly when adding several hundred virtual requests in each round of assignment simulation. Out proactive rebalancing method will also help guide vehicles to the locations of potential future requests, but it does not have the above two disadvantages.

Since the demand is sparsely distributed on thousands of network nodes in a real-size network, it is hard to predict the future demand at the node level. As we only need to guide the vehicles to areas where requests are likely to appear, we use the K-means clustering algorithm to spatially cluster the nodes

according to their geo-coordinates. We assume that each passenger has a walking range of α miles. Then, the number of clusters k is determined by satisfying $\frac{\text{Total area}}{k} \approx 2\pi\alpha^2$. Based on historical data, we build the probability distribution $P(n \mid o, \xi)$, which is the probability of n requests appearing in cluster o given a time interval ξ. A time interval ξ is defined by a tuple (t_s, t_e) where t_s and t_e are the start time and the end time of the time interval respectively. Let n_o^ξ be the maximum number of requests appearing in cluster o at time interval ξ according to historical data. We can generate $\sum_{o \in O} n_o^\xi$ virtual requests for time interval ξ, where O is the set of clusters. For each virtual request r, we set the probability of it appearing at time interval ξ, which is denoted by p_r, as follows.

Assume that we are considering cluster o and time interval ξ. The virtual requests are denoted by r_1, r_2, ..., $r_{n_o^\xi}$. As the definition reveals, $P(n \mid o, \xi)$ is the joint probability of exactly n requests occurring among all n_o^ξ requests. What we want to use is the marginal probability p_{r_i} for $1 \leq i \leq n_o^\xi$. We cannot solve all these marginal probabilities because the correlations between the requests are unknown. To help get a solution, we assume that r_i can only appear when all requests in the set $\{r_j \mid 1 \leq j < i\}$ appear. This assumption implies the order on these requests, i.e., when there are i requests appearing, they would be r_1, r_2,..., r_i. Under this assumption, we derive that $p_{r_i} = \sum_{n=i}^{n_o^\xi} P(n \mid o, \xi)$.

We can use formulation in Section 2.2 considering the set of virtual requests with a probability higher than $p_{\min}$[4]. The number of virtual requests can be large when the demand is high. To reduce the scale of the optimization problem, we ignore the virtual requests if there exists an idle vehicle within a predefined travel time range[5]. In summary, this method has the following advantages: i)

[4]We let $p_{\min}$ be 0.75 in the numerical experiments.

[5]We let the predefined range to be half of the maximum waiting time in the numerical experiments.

Compared to the predictive method in [3], our proactive rebalancing method requires predictions of only the origins instead of origin-destination (O-D) pairs. The probabilistic prediction is under less noise. Since we only consider the requests with a high probability, we are more confident that the vehicles are guided to areas with potential future requests; ii) The method does not increase the scale of the trip-vehicle simulation; iii) Since we proactively send idle vehicles to areas where future requests appear, the waiting time and delay for those requests can be potentially decreased.

2.3 Numerical experiments

2.3.1 Experimental setup

We use taxi trip data in Manhattan, NYC from 6 am to 12 am on an arbitrary day (Monday, May 6th, 2013) [25] as the travel demand. The network we use is the entire road network of Manhattan (4092 nodes and 9453 edges) [1, 97]. The link travel time is the daily mean travel time, which is computed using the method in [97]. The shortest paths between every two nodes in the network are precomputed and stored in a look-up table. We evaluate the performance of the techniques by comparing the metrics of interest in varying cases (different fleet sizes, capacities, maximum waiting time, maximum delay) between three algorithms: i) the framework in [1], denoted by *original*; ii) our framework with the speed-up techniques[6], denoted by *speed-up*; iii) our framework with the speed-up techniques and the proactive rebalancing algorithm, denoted by

[6]Note that the rebalancing formulation 2.8 is also used in the *speed-up* framework.

Table 2.1: Algorithm performance comparison.

Number of vehicles	Capacity	Ω (s)	Δ (s)	Algorithm	Computation time (s)	Service rate	Waiting time (s)	Total delay (s)
1000	4	120	240	Original	4.56	0.445	73.58	132.82
				Speed-up	0.78	0.461	73.67	132.43
				Speed-up+Proactive	1.02	0.460	72.91	131.13
2000				Original	8.10	0.697	71.70	120.70
				Speed-up	1.38	0.768	72.11	117.51
				Speed-up+Proactive	2.41	0.762	70.41	114.57
3000				Original	10.37	0.785	69.37	110.18
				Speed-up	1.92	0.904	67.62	95.84
				Speed-up+Proactive	4.28	0.913	64.66	87.81
1000		300	600	Original	45.46	0.608	172.54	377.55
				Speed-up	2.92	0.609	171.28	373.78
				Speed-up+Proactive	2.61	0.609	171.30	373.70
2000				Original	58.26	0.950	147.79	305.12
				Speed-up	5.04	0.957	146.97	301.02
				Speed-up+Proactive	4.79	0.956	143.78	293.73
3000				Original	60.14	0.988	119.87	200.15
				Speed-up	5.25	0.992	116.53	185.97
				Speed-up+Proactive	6.02	0.996	95.59	131.64
1000	10	120	240	Original	5.20	0.454	73.14	132.86
				Speed-up	0.76	0.467	73.55	132.42
				Speed-up+Proactive	1.03	0.464	72.71	131.16
2000				Original	8.85	0.701	71.61	120.67
				Speed-up	1.37	0.769	72.07	117.34
				Speed-up+Proactive	2.43	0.763	70.53	114.82
3000				Original	11.24	0.793	69.07	109.42
				Speed-up	2.81	0.904	67.80	96.19
				Speed-up+Proactive	4.26	0.913	64.63	88.26
1000		300	600	Original	120.75	0.661	158.26	373.29
				Speed-up	2.81	0.667	161.17	372.05
				Speed-up+Proactive	2.63	0.659	160.32	369.98
2000				Original	80.39	0.960	140.30	297.76
				Speed-up	4.85	0.963	143.61	299.02
				Speed-up+Proactive	4.80	0.962	139.85	289.87
3000				Original	49.61	0.989	117.27	198.59
				Speed-up	5.25	0.991	115.47	184.88
				Speed-up+Proactive	5.95	0.996	95.19	131.33

speed-up+proactive. The system is implemented using Python 3.5, and all experiments are conducted on a 4 core 3.4GHz computer. The maximum waiting time and the maximum total delay are assumed to be the same for all requests, which are denoted by Ω and Δ respectively.

2.3.2 Experimental results

We conduct two sets of experiments with a fleet of capacity 4 vehicles and a fleet of capacity 10 vehicles respectively. During the experiments, we collect the following metrics: the mean computation time (average computation time for a round of assignment simulation), service rate, mean waiting time and mean total delay. These metrics are shown in Table 2.1.

We first analyze the performance of the *speed-up* framework. The speed-up techniques decrease the computation time significantly in all cases with no optimality loss. Specifically, the average computation time reduction is as high as 87.46%, while the mean waiting time and mean in-vehicle delay remain at the same level. The maximum computation time reduction is 97.67% when operating a fleet of 1000 capacity 10 vehicles with $\Omega = 300$ and $\Delta = 600$. When Ω and Δ increases, the computation time of the *original* framework increases significantly, but the computation time of our *speed-up* framework remains at the same magnitude. In addition, the service rate is increased by 3.5% on average. For our experiment time span, this represents that 4068 more passengers are served in 6 hours. Two potential reasons for this to happen: i) the search space pruning technique discussed in Claim 3 allows us to use less time to conduct the exhaustive search to check a trip's feasibility. Therefore, we can explore more feasible solutions in the RTV-graph given a computation time limitation[7]. ii) the rebalancing formulation 2.8 is more efficient due to the newly added constraints. The maximum gap (11.9%) happens when operating a fleet of 3000 capacity 4 vehicles with $\Omega = 120$ and $\Delta = 240$.

The *speed-up+proactive* framework offers a 81.44% computation reduction on average compared to the *original* framework. The average computation reduction is slightly lower than the *speed-up* framework because the number of virtual requests can be higher than the number of unassigned requests. However, the service rate is increased by 4.8% on average, which is higher than the *speed-up* framework. In addition, the waiting time and total delay are decreased by 5.0% and 10.7% respectively. The largest gap happens when operating a fleet of 3000 capacity 4 vehicles when $\Omega = 120$ and $\Delta = 300$, where the *speed-up+proactive*

[7]In [2], when computing the RTV-graph, a time limit to explore potential trips per vehicle is imposed.

framework reduces the waiting time and the total delay on average by 24.28 seconds and 68.51 seconds respectively.

CHAPTER 3

A FRAMEWORK TO INTEGRATE MODE CHOICE IN THE DESIGN OF MOBILITY-ON-DEMAND (MOD) SYSTEMS

3.1 Introduction

The rapid growth of Mobility-on-Demand (MoD) services such as Uber and Lyft is disrupting urban mobility. Over the last few years, this disruption has led to much interest in studying the design, management and impacts of these systems. Several analytical and simulation-based studies have focused on the efficient design of the MoD system – allowing passengers to share rides and managing the fleet to serve travel demand with the minimum fleet size, passenger's waiting time, and vehicle miles traveled. In one of the first studies to develop a practical framework for managing a large-scale MoD fleet, [69] devised a heuristic-based taxi dispatching strategy and fare management system that could handle operations of large urban scaled fleets. To incorporate ridepooling (capacity 2), [97] developed the notion of a shareability graph, and showed that almost all the taxi demand in Manhattan, New York (around 150 million annual trips as of 2011) could be served by pairing up requests (2 requests per taxi). Pushing this result further, [1] presented a computationally efficient *anytime optimal* algorithm that can enable real-time high capacity ridepooling (up to 10 riders per vehicle) at the scale of Manhattan. Results show that 98% of the taxi demand in Manhattan can be served by 3000 vehicles of capacity 4 instead of the current fleet of 13586 active taxis.

With recent advancements in automation technologies and government reg-

ulations enabling this technology, autonomous MoD (AMoD) systems[1] are also gaining research interest [58, 102]. For example, an agent-based simulation of AMoD services are conducted for both Austin, Texas [31] and Melbourne, Australia [22]. Both studies concluded that AMoD services can satisfy travel demand with around one-tenth of privately-owned vehicles. A few simulation-based studies also subsequently quantified the impact of MoD and AMoD services on the environment [29], congestion [36], and parking [122].

In areas where public transit serves a large proportion of demand, the interaction between MoD services and transit should not be overlooked. Considering MoD services as a complementary mode of transit, a few studies have designed a bimodal MoD system to solve the first and last mile problem [70, 79, 100, 112]. [70] developed a real-time dispatching policy for MOD services in cooperation with existing mass transit service and tested it on a real network between Luxembourg City and its French-side cross-border area. [100] studied the integration of MoD service with Singapore transit using agent-based simulation. The authors concluded that the integrated system is financially viable and has the potential to make transit attractive by reducing out-of-vehicle time.

However, all these studies assumed a fixed and exogenous demand for MoD services. Most models assume a waiting time threshold with the demand that cannot be picked up within that threshold being considered *unsatisfied* or *drop-out* demand. The inherent assumption here is that passengers (from a certain demand set) that can be served by MoD will necessarily choose it. In practice, this may not be true: MoD services coexist with other travel modes (e.g., tran-

[1]Note that the design framework of MoD and AMoD is similar if behavioral aspects and driver economics are omitted.

sit and walking). The demand for MoD is not only a function of its attributes (e.g., price, travel time), but also depends on characteristics of the competing travel modes as well based on individual preferences. Therefore, even if a passenger can be served by MoD services, they may choose other travel modes for a number of reasons.

Moreover, optimizing the operational performance of AMoD or MoD systems under an exogenous demand cannot answer policy questions related to the extended impact of these services on the urban transportation system as a whole. Such questions include the impact of MoD services on transit ridership [6], vehicle ownership [46], and demand for parking [121]. Only a handful of studies have considered an endogenous demand model where MoD or AMoD systems compete with other travel modes based on service quality. [50] and [6] integrated such models into the MATSim and SimMobility platforms which are multi-modal agent-based activity simulators. However, both simulators suffer from a common problem of relying on the scenario-based supply-demand analysis of virtual cities, not optimizing the supply-side parameters, and using a synthetic mode choice model.

A few recent studies have considered supply-demand interactions in a multimodal transportation system in the presence of on-demand mobility service. These studies have done so in the context of agent-based stochastic user equilibrium with a day-to-day adjustment process [24]. In particular, [23] models the dynamic adjustment process of an MoD operator. However, unlike our design-focused approach, these studies emphasize evaluating the sensitivity of different MoD operating policies as opposed to determining the optimal supply-side response.

The contribution of this study is twofold. The first contribution is to develop a unified framework which integrates mode choice models with a state-of-the-art system for modeling real-time on-demand mobility services with varying passenger capacities (based on [1]). We illustrate the proposed framework by developing a multi-scale MoD fleet management system to serve the taxi demand in Manhattan where passengers are utility maximizers and choose a travel mode from a set of four mutually exclusive alternatives: ride-hailing service (capacity 1), ridepooling service (capacity 4), micro-transit service (capacity 10) and public transit (e.g. subway). The first three travel modes are MoD services assumed to be run by a single MoD provider. Note that the underlying mode choice model is calibrated with stated preference survey data from New York City.

The second contribution of this study is to implement and integrate a Bayesian Optimization (BO) based solver to find the optimal supply-side MoD parameters (e.g., fleet size, fare, etc). Historically, two types of methods have been used to determine the supply-side parameters of MoD systems: i) Heuristic methods to simultaneously optimize supply-side parameters and the vehicle routing problem (VRP) [63, 92]. Due to the complexity of VRP, these methods are inefficient for optimizing large-scale MoD operations; ii) Predefine candidate combinations of parameters (or scenarios), and then conduct a grid search to find the one that gives the optimal objective function value in the simulation [5, 11, 13, 61, 73]. The performance of this approach highly depends on the quality of the predefined set of candidate solutions. In addition, grid search struggles in finding even near-optimal solutions in a high dimensional space in reasonably short time. In this work, we decouple the optimization of supply-side parameters and the MoD vehicle assignment problem. We consider the

transportation simulation system as a black-box function, and use BO as a sequential search strategy to optimize the supply-side parameters.

The rest of this chapter is structured as follows. In Section 3.2, we formulate the mathematical problem and propose the unified framework. In particular, we describe two essential components: the mode choice model and the MoD simulation system. In Section 3.3, we introduce a BO-based approach to find the optimal supply-side parameters. Section 3.4 consists of numerical experiments on the Manhattan network, which are focused on the numerical convergence of the proposed framework, efficiency of the BO approach introduced in this work, and practical insights that the framework provides.

3.2 Framework: Modeling Mobility-on-Demand Systems within a Multimodal Transportation System

We consider a general transportation system that includes public transit and three classes of MoD services with varying passenger capacities (i.e., maximum occupancy). The public transit option is considered to be a complimentary travel mode to the MoD services, but note that the framework is not limited to this specific case[2]. The goal is to develop a simulation optimization framework to optimize both i) the system-level objective functions (e.g., MoD system operator's profit or consumer surplus) when the demand for MoD services is not exogenous and rather depends on their and other competing travel modes' level of service (e.g., waiting time, travel time, and travel cost), and ii) the trans-

[2]In the framework, the interplay between all travel modes is considered in the mode choice model. After the demand for each travel mode is generated, the simulator for each mode is run independently. Therefore, we can use any set of candidate travel modes for any purposes.

portation system operations given a certain set of supply and demand side parameters. The proposed simulation and optimization framework is shown in the Figure 3.1.

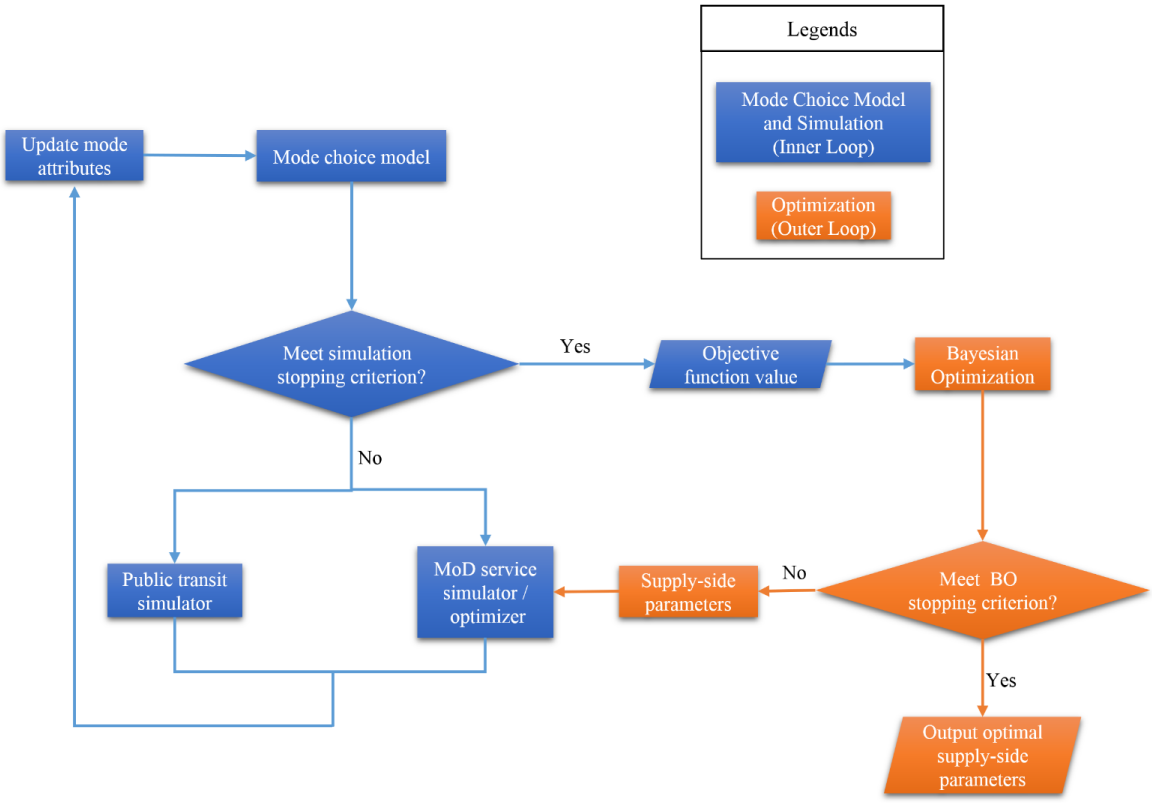

Figure 3.1: A framework to optimize the supply-side parameters for MoD system with the integration of the mode choice model.

The proposed framework can be disentangled into inner and outer loops. The outer loop iteratively optimizes the supply-side parameters using BO and terminates when a stopping criterion of BO is satisfied. For a given set of supply-side parameters, the objective function (e.g., operator's profit) of the outer loop is evaluated at the equilibrium of the inner loop. The inner loop iteratively performs the following steps: i) evaluate mode choice probabilities, ii) simulate the demand for and operations of public transit and MoD services, iii) update mode-specific attributes (e.g., waiting time) as the inputs to the choice

model in the next iteration, and iv) repeat until an equilibrium is reached, i.e. when the average difference in mode shares of two consecutive iterations is lower than a predefined threshold (see Section 3.4.1). In other words, if each iteration of the inner loop is interpreted as a "day", for a given set of supply-side parameters, passengers learn from the experienced historical level-of-service of travel modes so that they can make more informed mode choices on a given day. The inner loop terminates when passengers saturate their learning and make nearly consistent mode choices on consecutive days.

This section discusses the different components of the inner loop such as the mode choice model, and operations of public transit and MoD services. Section 3.3 discusses a BO-based approach to find the optimal supply-side parameters (outer loop).

3.2.1 Mode Choice Model and Data

To understand preferences of New Yorkers for MoD services, we conducted a stated-preference (SP) study. In an online survey, the respondents were asked about sociodemographics, travel characteristics, and various other opinions. The survey also had a discrete choice experiment (DCE) in which each respondent was asked to choose the best and the worst travel mode from a set of three choices: Uber (without ridesharing), UberPool[3] (with ridesharing), and their current travel mode (the one used most often on their most frequent trips). Table 3.1 shows the attribute levels of the DCE design. In the case of monthly payment of trip and parking costs, a per trip cost was computed by dividing monthly cost with trip frequency. The per mile cost of a private car ($.45) was

[3] *UberPool* represents MoD services with a passenger capacity of more than one.

obtained by summing the insurance, maintenance, and fuel cost (AAA News Room, 2016). In the DCE design, we made sure to constrain in-vehicle travel time (IVTT) and per mile cost of Uber to be less and more, respectively, than those of UberPool. Respondents were provided textual and visual information about all alternatives at the starting of DCE.

Table 3.1: Experiment Design for Mode Choices.

	Uber (Without Ridesharing)	**UberPool (With Ridesharing)**	**Current Mode**
Walking and Waiting Time	25%, 50%, 75%, 100%	25%, 50%, 75%, 100%	asked (100%)
In-vehicle Travel Time	80%, 95%, 110%, 125%	90%, 105%, 120%,135%	asked (100%)
Trip Cost Per Mile ($) (Excluding Parking Cost)	0.55, 0.70, 0.85, 1.0, 1.2	0.45, 0.60, 0.70, 0.80	asked or computed
Parking Cost	0	0	asked
Powertrain	Gas, Electric	Gas, Electric	Gas
Automation	Yes, No	Yes, No	No

Note: All % are relative to the reference alternative.

To obtain priors for pivot-efficient DCE designs[4], we first conducted a pretest. A D-efficient design with zero priors containing 4 blocks (6 choice situations per block) was generated with the Ngene software [20] for the pilot study. The attribute levels of Table 3.1 were used in the design. The online pilot survey was created using the web-based Qualtrics platform, but the survey was distributed among a continuous panel provided by Survey Sampling International (SSI, a professional survey firm) in February 2017. Those who drive for any MoD service or are younger than 18 years were considered ineligible to participate in the survey. Those who completed the survey in less than 10 minutes or provided conflicting responses (e.g., reported more children than household size) were discarded. After eliminating such responses, 298 (out of 397) completed responses were used as valid pretest observations for further discrete choice analysis.

We used prior parameter estimates from the pilot study to create a pivot-

[4]In pivot-efficient designs, attribute levels shown to the respondents are pivoted from reference alternatives for each respondent. In this study, the travel mode used on the most frequent trips was considered as the reference alternative.

efficient design with 6 blocks (7 choice situations per block). All the attributes and attribute levels remained the same (Table 3.1). Table 3.2 shows an example of the choice situation of the final mode choice experiment.

Table 3.2: An Instance of the Choice Experiment

	Uber (Without Ridesharing)	**UberPool (With Ridesharing)**	**Current Mode: Car**
Walking and Waiting time	6 minutes	9 minutes	12 minutes
In-vehicle Travel Time	38 minutes	50 minutes	48 minutes
Trip Cost (Excluding Parking Cost)	$11	$8	$6
Parking Cost	–	–	$6
Powertrain	Electric	Gas	Gas
Automation	Service with Driver	Automated (No Driver)	–

We conducted the main study during October-November 2017. After data validation tests, preferences of 1507 (out of 1689) respondents were used in the model estimation. We estimated a multinomial logit (MNL) model and used MNL attribute valuation for prediction of mode choice probabilities, which are needed in the simulation. The closed-form choice probability MNL expressions allow a seamless integration in the MoD simulation framework[5]. Table 3.3 summarizes MNL parameter estimates. By dividing marginal utilities of attributes with that of trip cost, willingness-to-pay (WTP) estimates are derived. By using these marginal rates of substitution, the model provides evidence that New Yorkers are willing to pay $25.9 and $18.6 to save an hour of OVTT and IVTT, respectively. The recommended hourly value of travel time for local commute by passenger car in downstate New York is $15.6 (Department of Transportation, New York State, 2012), which is close to our estimate of WTP to save an hour of IVTT. Another study of economic evaluation (US Department of Transportation, 2011) estimates that walking, waiting, and transfer time in personal travel should be valued at $19.10 - $28.70 per hour. Our estimates of WTP to save an hour of OVTT falls in this range. Although WTP estimates can be transferred

[5]We tried nested logit specifications with different nesting structures, but in all scenarios, one of the nests' elasticity estimates was above one. In other words, nest correlation was out of the accepted range $[0, 1]$ for compatibility with random utility maximization. This implies preferences of the sample are not aligned with nested logit correlation.

directly to the MoD simulation, alternative specific constants (ASCs) require recalibration (ASCs in SP studies are just manifestation of sample shares of alternatives). This process is consistent with the estimated mode choice model, but is not ideal because choice sets of SP study and the considered taxi demand are different[6]. Details about recalibration of ASCs can be found in Sections 3.4.2 and 3.4.4.

Table 3.3: Multinomial logit model estimates

Attributes	**Coef.**	**Std. Err.**	**Z-stat**
Walking and waiting time (OVTT, in mins)	-0.032	0.0020	-16.3
In-vehicle travel time (IVTT, in mins)	-0.023	0.0025	-9.3
Trip cost	-0.074	0.0030	-24.3
Parking cost	-0.057	0.0091	-6.3
Electric	-0.041	0.0386	-1.1
Automation	-0.182	0.0409	-4.5
ASC			
Uber	-0.821	0.0646	-12.7
UberPool	-1.266	0.0519	-24.4
Carpool	0.057	0.1594	0.4
Ridesharing	-1.689	0.1592	-10.6
Transit	-0.232	0.0491	-4.7
Car		base	
Loglikelihood		-9762.281	
Number of observations		1507	

3.2.2 Simulation of MoD and Transit Operations

In this section, we introduce the system we use to simulate the operations of the MoD and transit services. The simulation of the MoD services follow the approach discussed in Section 2.1.3.

[6]The recalibration of ASCs is much easier for the full travel demand model where all possible alternatives are available to travelers and their real mode shares are known.

Simulation of the Public Transit Service

We assume that the network and operational characteristics of the existing public transit system (headway, travel time, etc) are known. To obtain the level of service attributes of the public transit in the mode choice model, we perform a transit network assignment for each origin-destination (OD). We first combine the road network we use in MoD service simulator and the public transit network. Then we use an all-pair shortest path algorithm to find the walk-transit-walk path for each OD pair. This procedure consists of the following steps:

(1) Construct the public transit network. The public transit data is obtained from the General Transit Feed Specification (GTFS) dataset published by the transit authority through Google's GTFS project. The public transit network topology is created according to the schedules and associated geographic information in the GTFS data. The weight on each link is the scheduled travel time between two transit stations.

(2) Combine the road network and the public transit network. The weight on the link between two road network nodes is set to the walking time (distance/walking speed). For each road network node, we add a link to connect it to each public transit network node within the walking range (e.g. 0.5 miles). As the origins and destinations are part of the nodes in the road network, the weight on the link connecting the road network node and the public transit node is set to a generalized cost, which is the sum of the walking time (distance/walking speed), the expected waiting time (half of the headway for the public transit line) and the trip fare (converted to seconds) for the public transit line.

(3) Compute the all-pairs shortest path using the combined network, and store them in a look-up table. During the simulation, we refer to the table to find the path and mode-specific attributes for public transit in the mode choice model.

3.2.3 Updating Historical Trip Attributes and Inner Loop Stopping Criterion

To model the memory and learning process of the travelers, we store and update the historical trip attributes for each OD after each iteration of the inner loop, and use these values as inputs to the mode choice model at the next iteration. Since the demand is sparsely distributed on the thousands of network nodes (e.g., 4092 nodes in our network), we use K-means clustering algorithm [71] to spatially cluster the nodes. If the walking range for the passenger is r miles, we determine the number of clusters k by satisfying $\frac{\text{Total area}}{k} \approx 2\pi r^2$. The historical trip attributes are stored and updated at the cluster-level.

Due to the maximum waiting time Ω and the maximum delay Δ constraints, some of the demand for a certain MoD service may not be satisfied by the system. We assume that the demand that is unsatisfied (due to these constraints) will switch modes and take public transit. Therefore, for each cluster pair (i, j), we also store the service rate $s_{i,j,m}$ for each MoD mode m, which is the proportion of travel demand served by mode m for cluster pair (i, j).

We compute the utility of the MoD services for the next iteration as follows (accounting for unsatisfied requests): Assume that for the cluster pair (i, j), the

utility of taking MoD mode m and being picked up in the simulation is $U'_{i,j,m}$, and the utility of taking transit is $U_{i,j,t}$. Then, the utility of taking the MoD mode m in the next iteration $U_{i,j,m}$ is computed according to the following equation:

$$U_{i,j,m} = s_{i,j,m} \cdot U'_{i,j,m} + (1 - s_{i,j,m}) \cdot c_m \cdot U_{i,j,t} \tag{3.1}$$

A constant penalty multiplier c_m is used since transit was not the passenger's first choice in the last iteration. A high c_m penalizes the utility function more if a request is not satisfied, and further decreases the corresponding MoD mode's share. In the numerical experiments, we use $c_m = 2$. This parameter should ideally be calibrated using a stated preference study, which we leave for the future research. $U'_{i,j,m}$ and $U_{i,j,t}$ can be computed according to the historical trip attributes. After each iteration, the historical trip attributes are updated according to the following equation:

$$\vec{H^{n+1}_{i,j,m}} = \beta \vec{H^{n}_{i,j,m}} + (1 - \beta) \vec{I^{n}_{i,j,m}} \tag{3.2}$$

where $\vec{H^{n}_{i,j,m}}$ is the set of historical trip attributes for cluster pair (i, j) for mode m at iteration n which consists of the in-vehicle travel time, waiting time and the service rate, β [7] is a constant coefficient that controls the balances between the historical information and the new information, $\vec{I^{n}_{i,j,m}}$ is the trip attributes for cluster pair (i, j) for mode m which we obtain from the simulation in the current iteration by averaging the attributes for all ODs in the cluster pair.

The inner loop simulation procedure is iterated until the following stopping

[7] In our numerical experiments, we let $\beta = 0.5$.

criterion is satisfied:

$$Z^n = \frac{1}{|M|} \sum_{m \in M} |S_m^n - S_m^{n-1}| \tag{3.3}$$

where Z^n represents the average mode share difference among all the travel modes between iteration n and $n-1$, M is the set of candidate travel modes, S_m^n is the share of mode m at iteration n. If Z attains a value smaller than a predefined threshold of 0.01, the inner loop is terminated. In other words, the learning of travel demand saturates and system appears to achieve an equilibrium as a result of their consistent travel mode choices on consecutive "days".

3.3 System Optimization using Bayesian Optimization

In Section 3.2, we discussed a simulation framework where the inner loop considers the supply-demand interaction for a given set of operational parameters of MoD services. In this section, we study the problem of optimizing these parameters which include fleet size and the pricing rules for MoD services with varying passenger capacities.

The inner-loop simulation can be considered as a black-box function $f(\vec{x})$ for which the set of supply-side parameters $\vec{x}$ (i.e., decision variables) should be chosen such that the system performance metric at the equilibrium of inner loop attains its optimal value. The performance metric depends on the influence of different stakeholders. For example, whereas policy-makers might be interested in optimizing the consumer surplus and vehicle miles traveled, MoD operator might be interested in just maximizing the profit.

This optimization problem is difficult to solve because of the following properties:

- The objective function cannot be evaluated analytically and its derivatives are thus not easily available;
- One can obtain observations of the objective function, but the function evaluation for even one set of decision variables is computationally expensive;
- The function evaluation can be affected by simulation noise such as prediction of mode choice probability and other random factors (e.g. the initial location of the vehicles) in the MoD service simulator.

Whereas these characteristics make the problem intractable to solve using analytical optimization methods, Bayesian Optimization (BO) is an appropriate and powerful tool in such settings. BO is a sequential search strategy for the global optimization of an expensive black-box function $f(x)$ [78]. It first emerged as a successful strategy in many machine learning applications [9,10,101,106,107,117], and has lately been employed in many other areas including robotics [14,67], sensor networks [40], environmental monitoring [72], information extraction and retrieval [59,114], and game theory [88].

Since the objective function does not have a closed-form expression in terms of the decision variables, BO treats it as a black-box function with some prior belief. Here *prior* represents our belief about the space of possible objective functions. As we obtain more observations of the function, our belief about the objective function is updated by combining the prior and the likelihood of the data already acquired to get a potentially more informative posterior. The

posterior distribution is then used to select the next set of decision variables for the function evaluation.

Specifically, a BO framework has two key ingredients: i) a probabilistic surrogate model for the expensive objective function f; ii) an acquisition function that is maximized to determines the next point for function evaluation [99]. The BO framework is demonstrated in Algorithm 1, where S is the surrogate model, α is the acquisition function, and $\mathcal{D}_i = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_i, y_i)\}$, which is the set of historical function observations until iteration i. Each observation is denoted as $(\vec{x}_i, y_i)$ where $\vec{x}_i$ is the set of inputs, and y_i is the objective function value we obtain by evaluating the objective function f.

Algorithm 1 Bayesian Optimization

1: **for** $i = 1, 2, ...,$ **do**
2: $\quad \vec{x}_i = \underset{\vec{x}\alpha(\vec{x}, \mathcal{D}_{i-1})}{\text{argmax}}$
3: $\quad$ Evaluate the objective function using $\vec{x}_i$ to get y_i
4: $\quad \mathcal{D}_i = \mathcal{D}_{i-1} \cup (\vec{x}_i, y_i)$
5: $\quad$ Update S.
6: **end for**
7: Output the best y.

We now provide a brief introduction of these two ingredients. Readers can refer to [12] and [99] for more details of the method.

3.3.1 Surrogate Model

Since evaluating $f(x)$ is expensive, the BO framework builds a surrogate model using the historical observations to approximate the objective function. In this work, we adopt the Gaussian Process (GP) as the surrogate model, which is the most commonly-used model in BO [77]. GP is a stochastic process which can be

seen as a collection of random variables where any finite set of random variables follows a multivariate Gaussian distribution. Assume that the set of points we use to build the GP is denoted by $\mathcal{X}$, then the GP is a fully specified by its mean function $m : \mathcal{X} \to \mathbb{R}$ and covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Intuitively, GP is a distribution over functions. Given a point $\vec{x}$, it will return the mean and the variance of a normally distributed variable y which is a prediction for $f(\vec{x})$.

For simplicity, the mean function $m(\cdot)$ is usually defined to be zero function, i.e., $m(\cdot) = \vec{0}$ [12, 72, 114]. Assume that we have historical observations $\mathcal{D}_i$, and we want to use GP to predict $f(\vec{x^*})$ on an arbitrary point $\vec{x^*}$. GP will return a normally distributed variable y^* with both the mean and variance. Let $\vec{x}_{1:i} = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_i]^T$ and $y_{1:i} = [y_1, y_2, ..., y_i]^T$. As the definition of GP reveals, the joint distribution of the historical observations and y^* is as follows:

$$\begin{bmatrix} y_{1:i} \\ y^* \end{bmatrix} \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} \vec{K} & \vec{k}_{\vec{x^*}} \\ \vec{k}_{\vec{x^*}}^T & k(\vec{x^*}, \vec{x^*}) \end{bmatrix}\right) \tag{3.4}$$

where $\vec{k}_{\vec{x^*}} = [k(\vec{x^*}, \vec{x}_1), k(\vec{x^*}, \vec{x}_2), ..., k(\vec{x^*}, \vec{x}_i)]^T$, and $\vec{K}$ is the covariance matrix with each entry $\vec{K}_{(j,n)} = k(\vec{x}_j, \vec{x}_n)$.

Then, we can obtain the predictive distribution over y^* as follows:

$$y^* | \vec{x}_{1:i}, y_{1:i}, \vec{x^*} \sim \mathcal{N}(\mu(\vec{x^*}), \sigma(\vec{x^*})) \tag{3.5}$$

where $\mu(\vec{x^*}) = \vec{k}_{\vec{x^*}}^T \vec{K}^{-1} y_{1:i}$, $\sigma(\vec{x^*}) = k(\vec{x^*}, \vec{x^*}) - \vec{k}_{\vec{x^*}}^T \vec{K}^{-1} \vec{k}_{\vec{x^*}}$.

The choice of covariance function k for GP determines the smoothness properties of samples drawn from it [12]. Similar to many other studies, we use the

Matern kernel [75], which incorporates a smoothness parameter ς to provide flexibility in modeling functions:

$$k(\vec{x_i}, \vec{x_j}) = \frac{1}{2^{\varsigma-1}\Gamma(\varsigma)}(2\sqrt{\varsigma}\|\vec{x_i} - \vec{x_j}\|)^{\varsigma} H_{\varsigma}(2\sqrt{\varsigma}\|\vec{x_i} - \vec{x_j}\|), \tag{3.6}$$

where $\Gamma(\cdot)$ and $H_{\varsigma}(\cdot)$ are the Gamma function and the Bessel function of order ς, respectively. When $\varsigma = 1/2$, the kernel reduces to the unsquared exponential kernel, and when $\varsigma \to \infty$, the kernel reduces to the squared exponential kernel. To provide appropriate smoothness, BO applications usually use $\varsigma = 3/2$ and $\varsigma = 5/2$ [91]. In our experiments, we use $\varsigma = 5/2$. Readers can refer to [53] and [106] for more details about the GP as well as its covariance functions.

Acquisition Function

The acquisition function is used to select the next realization of the decision variable for the evaluation of the objective function. It usually considers both the mean and variance of the predictions provided by the surrogate model. Intuitively, the process can be seen as maximizing the utility of the next sampling.

A good acquisition function is the one that finds an elegant trade-off between exploration and exploitation[8] while searching for the optimum of the objective function. The BO literature proposes many acquisition functions such as the probability of improvement (PI) [56], the expected improvement (EI) [108], and the upper confidence bound (UCB) [105]. Similar to many other studies, we use UCB as the acquisition function.

[8]Exploration represents searching in regions with high uncertainty but might have observations with high objective function values. Exploration helps in avoiding being trapped in a local optimum. Exploitation represents searching in the regions with high expected values of the objective function given the information provided by historical function evaluations.

$$\text{UCB}(\vec{x}) = \mu(\vec{x}) + \kappa\sigma(\vec{x}) \tag{3.7}$$

where κ is a hyperparameter which establishes a balance between the exploration and the exploitation. In our experiments, we employ a special case of UCB which is called GP-UCB [14, 105] that casts the BO problem as a multi-armed bandit problem. In GP-UCB, κ is automatilly updated according to the following euqation:

$$\kappa = \sqrt{2\log(\frac{n^{d/2+2}\pi^2}{3\delta})} \tag{3.8}$$

where n is the number of past objective function evaluations, $\delta \in (0, 1)$ is a parameter of choice, d is the dimensionality of the search space. We use $\delta = 0.1$ the same as in [105].

3.3.2 Objective Function

In this work, we define the objective from the perspective of MoD service provider. Consider that all MoD operations are run by one service provider who is a profit maximizer. The profit is defined as the difference between the revenue and the operating cost. We use the ride-hailing service (capacity 1) as the base mode for the fare calculation of other MoD services and compute its trip fare according to the equation for UberX service [111]. The fare for all other MoD modes m with higher passenger capacities are computed by applying a discount factor γ_m ($0 \leq \gamma_m \leq 1$) on the fare of the ride-hailing service. For a passenger r that is served by the tier m MoD service, the trip fare f_r is computed

as follows:

$$f_r = (1 - \gamma_m) \cdot \max(f_{min}, f_{base} + f_t \cdot t_r + f_d \cdot d_r) \tag{3.9}$$

where f_{min} and f_{base} are the minimum fare and base fare for the ride-hailing service, t_r and d_r are the travel time and travel distance for r, f_t is the time rate for the ride-hailing service, which is the fare per second, and f_d is the distance rate for the ride-hailing service, which is the fare per mile.

The objective function $O(\boldsymbol{\gamma}, \vec{n})$ of the MoD service provider is computed as follows:

$$O(\boldsymbol{\gamma}, \vec{n}) = \sum_{i \in P} f_r - \sum_{m \in M} [(c_{m,l} + s_m) \cdot n_m + c_{m,d} \cdot d_m] \tag{3.10}$$

where P is the set of all passengers, $c_{m,l}$ is the leasing cost for a tier m MoD vehicle in the experiment time span, s_m is the salary that the operator pays to tier m driver[9], d_m is the total distance traveled by the tier m MoD service and is obtained from the simulator, and n_m and $c_{m,d}$ are the fleet size and operating cost ($ per mile) of the tier m MoD service. The set of discount factor and fleet size for each tier of MoD service are denoted as $\boldsymbol{\gamma}$ and $\vec{n}$, which are the decision variables.

Even though this work considers the objective from the perspective of a service provider, the proposed framework is more general and can be applied to designing policies, subsidies and regulatory strategies, guiding infrastructure planning and deployment, and operational management of transit services in the presence of MoD systems.

[9]The drive salary can be ignored in the case of AMoD systems.

3.4 Numerical Experiments

In this section, we present a series of numerical experiments to show: i) the numerical convergence to an equilibrium of the mode choices within the multimodal system; ii) the calibration of the ASCs; iii) the performance of the BO-based optimization algorithm; iv) relevant applications of the proposed framework. In the following experiments, *convergence to an equilibrium* represents that the system reaches a state where the average difference in mode shares of two consecutive iterations is lower than 1%. The system is implemented using Python 3.5, and all experiments are conducted on an Intel core I7 computer (3.4 gigahertz, 16 gigabytes RAM).

As a proxy for the real OD demands, we use the publicly available dataset of taxi trips in Manhattan, New York [26]. In our system, we serve this demand via either i) the ride-hailing service (capacity 1); ii) the ridepooling service (capacity 4); iii) the micro-transit service (capacity 10), and iv) public transit (e.g. subway). The first three modes are MoD services offered by one MoD service provider. We consider the pickup time for the taxi trips in the dataset to be their departure time . The network we use is the entire road network of Manhattan (4092 nodes and 9453 edges) [1, 97]. The link travel time is given by the daily mean travel time, which is computed using the method in [97]. For public transit, we use the subway network provided by the Metropolitan Transportation Authority [80]. The following experiments are conducted with respect to the rush hour demand (8 am to 9 am) on an arbitrary day (Monday the 6th of May, 2013).

The system has 5 decision variables: the fleet size of the ride-hailing service (n_1) and the fleet sizes and discount factors for the ridepooling and micro-transit

services (n_4, n_{10}, γ_4, and γ_{10}). Note that at the initialization stage, there is no information about the OD-specific MoD attributes (e.g., waiting time and travel time). Therefore, we initialize the attributes as follows: Let Ω be the maximum waiting time allowed in the MoD system and $t_{o,d}$ be the shortest path travel time for each (o, d). The travel time of the ride-hailing service, ridepooling service, and micro-transit service for (o, d) is set to $t_{o,d}$, $1.2t_{o,d}$, $1.5t_{o,d}$ respectively and the waiting time is set to 0.3Ω, 0.36Ω, 0.45Ω respectively. In our experiments, we set Ω at 10 minutes.

To account for the labor cost of the drivers, in this study, we consider a driver salary of \$17 per hour for all MoD services, which is close to the hourly mean wage of taxi drivers in New York state (United States Department of Labor, May 2017). The leasing cost includes insurance, depreciation, maintenance, and other registration charges, and the operating cost includes fuel and tier cost. We compute these costs for different MoD services using statistics provided by AAA News Room, 2016. The leasing cost of \$11.97 per day per vehicle is obtained for the ride-hailing and ridepooling fleet of Sedan cars and \$19.32 per day per vehicle for the micro-transit fleet of Minivans. We normalize these numbers based on the proportion of daily demand (5.94%) served during the experiment hour. The operating cost turns out to be \$0.1473 per mile for all MoD services.

3.4.1 Convergence to an Equilibrium

In this section, we illustrate the proposed multimodal system converging to an equilibrium with respect to mode choices using two test cases. In the test cases, we are not optimizing the system, but rather using a fixed set of supply-side

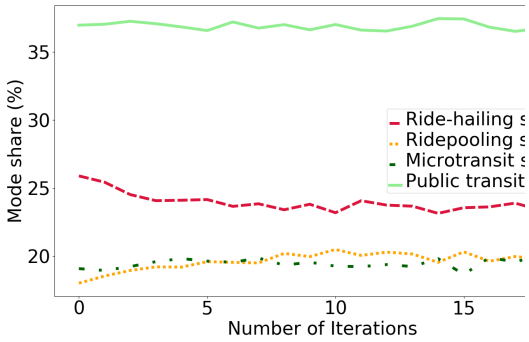

Figure 3.2: The mode share for the test case 1.

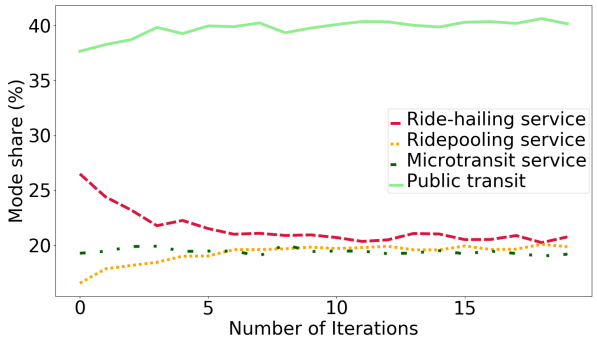

Figure 3.3: The mode share for the test case 2.

parameters to test for the numerical convergence of the inner loop simulation to an equilibrium. The experiments are run for 20 iterations for each set of parameters.

The supply-side parameters for the first case are as follows: $n_1 = 800$, $n_4 = 1000$, $n_{10} = 500$, $\gamma_4 = 0.2$ and $\gamma_{10} = 0.4$. The iterative variation in the share of each travel mode is shown in Figure 3.2. The share of each mode is relatively stable after around 5 iterations, which shows the numerical convergence of the system to an equilibrium. Note that the mode share is likely to fluctuate a little even after reaching equilibrium. This is just a manifestation of the simulation noise of the probabilistic mode choice model. In this test case, the mode share at equilibrium is not significantly different from the mode share at the beginning because the initial waiting time and travel time are close to the ones that the MoD services provider can offer given the supply-side parameters.

We set $n_1 = 100$, $n_4 = 1000$, $n_{10} = 50$, $\gamma_4 = 0.05$ and $\gamma_{10} = 0.4$ in the second test case. We intentionally consider the lower fleet size of the ride-hailing service (capacity 1) to make its level of service poorer than the one achieved with the initialized waiting time and travel time. The share of each mode is shown in

Figure 3.3. As expected, the mode share of the ride-hailing service quickly decreases in the first 5 iterations. According to initial conditions, the ride-hailing service is attractive but its demand rapidly decreases as passengers learn that the service is not able to serve the demand. In addition, the mode share of the ridepooling service increases rapidly in the first 5 iterations as it offers a higher level of service than initial conditions due to a high fleet size of 1000 vehicles. The stop criterion value for the last five iterations are reported. For test case 1, the stop criterion value for the last 5 iterations are 0.0031, 0.0029, 0.0018, 0.0027 and 0.0027. For test case 2, the stop criterion value for the last 5 iterations are 0.0045, 0.0025, 0.0027, 0.0028, and 0.0018.

Note that even with a fleet size of 100 vehicles in the second test case, the ride-hailing service could attain the larger share than both high capacity MoD services. There are two reasons for this to happen: i) The demand data of Manhattan contains many short trips. The passenger is likely to choose the ride-hailing service over high capacity services for short trips because even the discount cannot compensate for the lower level of service in high capacity MoD; ii) As discussed earlier in Section 3.2, we assume that unsatisfied MoD demand (when the waiting time is larger than the threshold) shifts to the public transit and the penalty on the utility of MoD service is applied in the next iteration. It appears that even though the service rate for ride-hailing service is low, the penalty is not high enough to shift the demand to other high capacity MoD services.

Additionally, the public transit share is very high (35% - 40%) in both cases, even though the input data corresponds to trips that were taken by taxi in reality. This is a manifestation of not having an alternative specific constant (ASC)

for transit in the mode choice models. Therefore, the ASC of the public transit needs to be calibrated to lower its mode share for this specific travel demand considered in this study, which we describe next.

3.4.2 Calibrating the Alternative Specific Constant of Transit.

In the absence of revealed preference data, calibration of ASCs is difficult. Since ride-hailing service share characteristics of Uber, and ridepooling and micro-transit services are similar to UberPool, we use their respective ASCs as our best guess in the case study. Moreover, MNL estimates of marginal utilities of alternative-specific attributes are used in the case study to ensure consistency of willingness to pay and scale.

If we had the true share of transit for the given demand, then the ASC of transit could have been directly calibrated using the method suggested by [110]. Since, the travel demand considered in the study corresponds to trips that actually used the taxi as travel mode, a transit mode share of 0-10% in our results was endogenously set as an appropriate target. To calibrate the transit ASC of the choice model for this specific population, we use a grid search approach. Since the demand we consider was served by 13,586 active taxis, we ran the inner loop simulation at different values of transit ASC with only transit and a fleet of 13,586 ride-hailing vehicles. The share for public transit for various ASC values are shown in Table 3.4. Transit attains a share of 6.3% at an ASC value of -3, which appears appropriate for the considered travel demand and thus is used in the remaining experiments. Figure 3.4 shows the results of running the test case 2 in Section 3.4.1 with -3 as the ASC for public transit. The shares of the

Table 3.4: ASC and resulting mode share for public transit.

ASC	Mode share (%)
-1.0	32.9
-1.5	23.2
-2.0	15.5
-2.5	10.2
-3.0	6.3

ride-hailing and the ridepooling services become 5.8% and 52.6% which were 20.8% and 19.9% when ASCs were not considered (see Section 3.4.1).

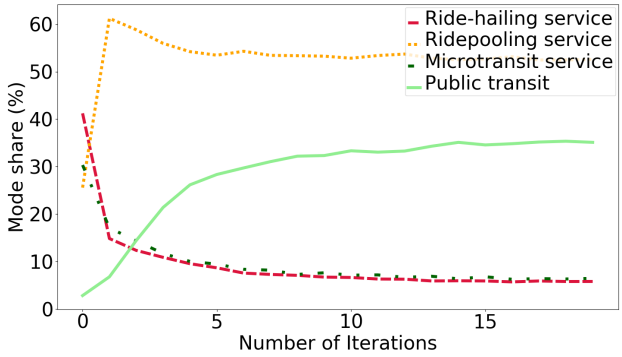

Figure 3.4: The mode share for test case 2 with the calibrated ASC of transit.

3.4.3 Algorithm Performance.

In this section, we illustrate the application of the proposed BO framework to optimize the supply-side parameters and evaluate its performance by: (1) Use a smaller test example (10% of the real travel demand) to compare BO solution with the brute-force solution; (2) Compare the performance of BO and random search method [8] using real-scale travel demand.

Optimizing the supply-side parameters for an MoD system is analytically intractable due to the complexity of the supply-demand interactions, since the fleet assignment problem is a function of the demand and vice versa. Therefore,

it is difficult to find an optimal solution even for relatively small test cases. Instead, we use the following approach to approximate the optimal solution via a grid search of relatively high resolution and compare that solution to that of BO. First we decrease the travel demand to 10%, and change the maximum value for n_1, n_4 and n_{10} to be 800, 300 and 150 respectively. The discount factors γ_4 and γ_{10} are set to constants (0.1 and 0.2, respectively) to further decrease the search space. In this setting, we enumerate every possible combination of the fleet size in $\{(n_1, n_4, n_{10}) \mid n_1 \in [25, 50, ..., 800], n_4 \in [0, 25, ..., 300], n_{10} \in [0, 25, ..., 150]\}$ and find the solution with the highest profit as the near-optimal solution. On the same test case, we employ BO with three different acquisition functions (UCB, PI and EI) to understand the sensitivity of BO's performance relative to the acquisition functions by comparing the gap between the respective BO solutions and the near-optimal solution found by the full enumeration method.

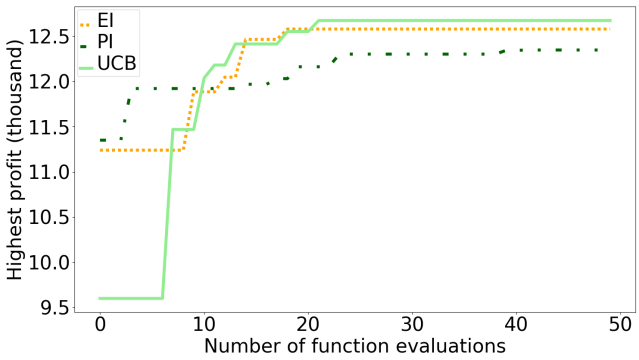

Figure 3.5: The comparison of the acquisition functions' performance.

In the full enumeration method, we evaluate the profit for 2912 parameter settings. The best solution is obtained with $n_1 = 225$, $n_4 = 150$, $n_{10} = 0$, and provides a profit of 13001. Figure 5 shows the variation in the highest profit with the number of function evaluations for BO using three different acquisition functions. It can be seen that PI performs worse than EI and UCB, while EI and UCB perform similarly well in our example. The best solution found by BO

is $n_1 = 182$, $n_4 = 85$, $n_{10} = 99$, which gives a profit of 12674. Although the optimal supply-side parameters provided by BO are quite different from the near-optimal solution of the full enumeration method, the profit gap is only about 2.5%.

In the second experiment, we use real-scale travel demand and compare the performance of BO and random search method. In the random search method, we sample the decision variables from a predefined distribution (uniform distribution in our case), evaluate the objective function at those realizations, and report the realization of the decision variable corresponding to optimal objective function value as the optimal solution. For a fair comparison, we keep the same number of function evaluations for BO and the random search method.

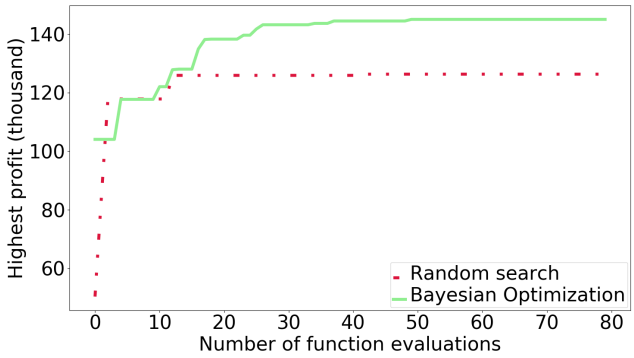

Figure 3.6: The comparison of the algorithms' performance.

Figure 3.6 shows the variation in the highest profit with the number of function evaluations in both optimization methods. The highest profit bound by BO and the random search are 145,015 and 126,308 respectively, which represents that BO's solution increases the profit by 15% compared to the random search's solution. This considerable gap between objective functions shows the advantage of using BO over the random search. We expect this gap to be larger in problems with more decision variables i.e., higher dimensional solution spaces.

The BO optimal solution is: $n_1 = 817$, $n_4 = 1539$, $n_{10} = 173$, $\gamma_4 = 0$ and $\gamma_{10} = 0$. It is important to note that the discount factor is 0 for both high capacity MoD services, which seems quite unrealistic because passengers are not likely to use high capacity MoD services if they have to pay the same price as of ride-hailing service. This experimental result implies that discounting the price of high capacity MoD services is not able to attract a level of demand that is high enough to compensate the reduction in profit due to the discounting. In general, the passengers have different perceptions about the discounted fare of the high capacity MoD services, but such latent factors are hard to quantify. The current mode choice model we use assumes that the trade-off between sharing a ride and the cost savings of doing so can be modeled as a linear relationship, and does not consider the fact that travelers perceptions of this trade-off may be more nuanced.

3.4.4 Scenario Analysis.

Modeling passenger's perception of discounting is left for future research but we illustrate implications of including such latent factors in choice models using a scenario-based analysis. We hypothesize that a passenger is less likely to choose high capacity MoD services at low values of the discount factor, even if these services have the same attribute-governed-utility[10] as of the ride-hailing service. We represent this hypothesis by adding a function of the discount factor, which is denoted as $f(\gamma)$, in the utility equation of the high capacity MoD services:

[10]The attribute-governed-utility implies the part of utility which depends on observed attributes of the travel mode such as travel time, waiting time and travel cost in our case.

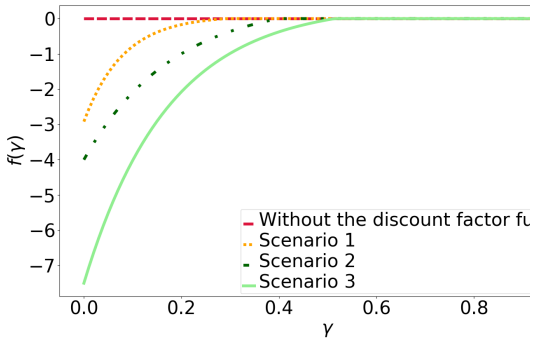

Figure 3.7: $f(\gamma)$ for the capacity 4 service.

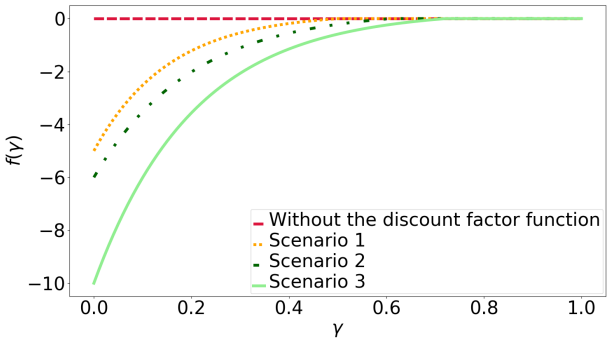

Figure 3.8: $f(\gamma)$ for the capacity 10 service.

Table 3.5: The discount factor functions for each scenarios.

Scenario	Discount factor function for the ridepooling service	Discount factor function for the micro-transit service
1	$0.08 - 3 \cdot e^{-12.4 \cdot \gamma}$	$0.2 - 5.2 \cdot e^{-6.5 \cdot \gamma}$
2	$0.5 - 4.5 \cdot e^{-5.5 \cdot \gamma}$	$0.3 - 6.3 \cdot e^{-5 \cdot \gamma}$
3	$0.4 - 7.9 \cdot e^{-5.8 \cdot \gamma}$	$0.3 - 10.3 \cdot e^{-4.9 \cdot \gamma}$

$$f(\gamma) = \min(0, a + b \cdot e^{-c \cdot \gamma}) \tag{3.11}$$

We call the function of the discount factor that we add to the utility equation *discount factor function*. We optimize MoD operations under three different discount factor functions, which are shown in Table 3.5. The coefficients in $f(\gamma)$ are set such that three scenarios represent low disutility, medium disutility and high disutility for having a low discount factor of high capacity MoD services. Since the passenger is likely to expect a higher discount for the micro-transit service than the ridepooling service, we use different values of coefficients in both functions to mimic these expectations. The plots of discount factor functions for the ridepooling and micro-transit services under three scenarios are shown in Figures 3.7 and 3.8.

Table 3.6: The solution for each scenario.

Scenario	n_1	n_4	n_{10}	γ_4	γ_{10}	Profit
With no discount factor function	817	1539	173	0	0	145015
1	2826	235	526	0.12	0.24	122838
2	2801	428	690	0.17	0.29	119154
3	2900	924	19	0.33	0.69	111141

Table 3.7: The mode share for each scenario.

Scenario	Ride-hailing share (%)	Ridepooling share (%)	micro-transit share (%)	Public transit share (%)
With no discount factor function	23.2	56.1	13.7	7.0
1	61.0	13.5	19.2	6.3
2	60.5	14.8	18.6	6.1
3	62.2	26.4	6.4	5.0

The optimal supply-side parameters under three scenarios were found using the BO-based approach and results are summarized in Table 3.6. The discount factors of all MoD services are positive after employing discount factor functions, and γ_{10} is (as expected) higher than γ_4. As the disutility of high capacity MoD services at low discount factors increases in scenario 2, the service provider has to offer more discount and increase the fleet size to attract the passengers towards high capacity services, and thus the profit also decreases. Note that the fleet size of micro-transit (capacity 10) is only 19 in scenario 3. This result represents the situation when the disutility of using micro-transit at a low discount factor is extremely high, and thus the service provider finds it hard to make profits by running this service.

Table 3.7 shows the share of each mode in all scenarios. In a base case scenario when discount factor function is ignored, a large portion of the demand (around 70%) is served by high capacity MoD services. However, in scenarios with a discount factor function, a large fraction of demand (around 60%) is served by the ride-hailing service. Therefore, the operating cost of the service

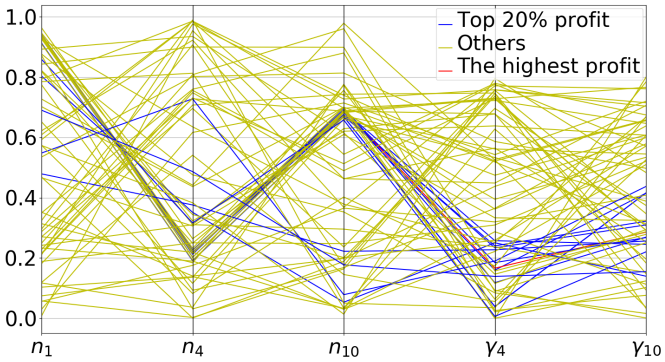

Figure 3.9: Decision variables at various function evaluations in the BO framework (scenario 2).

provider appears to increase in order to make the same revenue. An increase in the operating cost and a higher discount factor of high capacity MoD services result in a lower profit. Not only the share of micro-transit service decreases in scenario 3, its early demand shifts to the ridepooling service instead. This shift can be attributed to the similar values of $f(\gamma)$ for the ridepooling service in scenario 3 and for the micro-transit service in scenario 2 (see Figure 3.7 and Figure 3.8).

Figure 3.9 shows a parallel coordinates plot of decision variables for which the BO framework evaluates the objective function in process of searching for the optimal values in scenario 2. For visualization convenience, n_1, n_4 and n_{10} are normalized by dividing with their maximum values (3000, 2000 and 1000 respectively). The red, blue, and yellow lines represent the best, the top 20%, and all remaining solutions, respectively. The figure shows that the BO framework explores a broad domain of the solution space while exploiting the solution space with high expected profits more intensively.

In the experiments discussed above, we change both the discount factors and the fleet sizes for each MoD service to find the best solution using BO. It is dif-

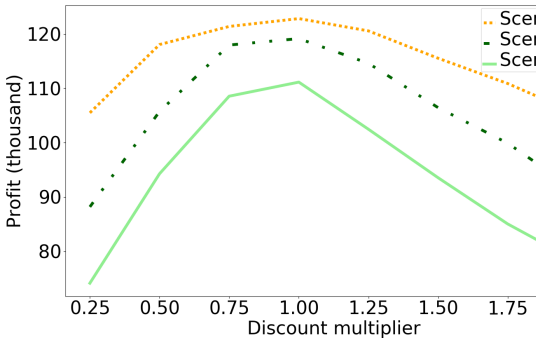

Figure 3.10: The profit for different discount multipliers and different scenarios.

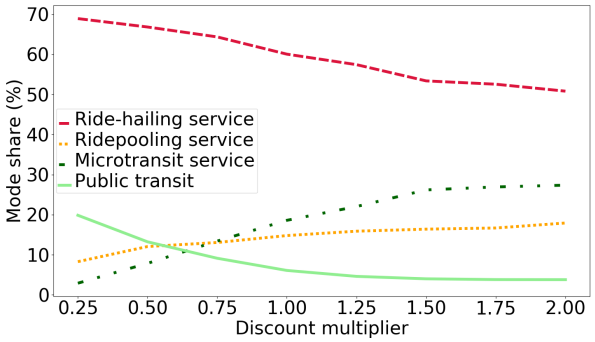

Figure 3.11: The mode share for Scenario 2 with different discount multipliers.

ficult to infer the effects of discount factors on the mode share and the profit of MoD service providers. Therefore, we conduct an additional set of experiments for each scenario mentioned above. In these experiments, the fleet size for each MoD service is kept constant, which is the solution reported in Table 3.6. We use different multipliers in $[0.25, 0.5, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00]$ to increase or decrease the discount factors reported in Table 3.6 for each scenario. For example, if the multiplier we used for Scenario 2 is 0.75, then $\gamma_4 = 0.75 \cdot 0.17 = 0.1275$. Figure 3.10 shows the profit for three scenarios with different discount multipliers. The profit first increases and then quickly decreases as the discount factors increase. The highest profit is given by the discount factors found by BO. Figure 3.11 shows the mode share for Scenario 2 with different discount factor multipliers. We should note that: (i) As the discount increases for MoD services, the mode share of public transit service keeps decreasing as it gradually loses the advantage of low cost; (ii) The mode share of ridepooling and micro-transit service increases as the discount factor increases, but the mode share of micro-transit service increases faster. The reason is that the base discount factor for micro-transit service is larger than ridepooling service, and thus the discount factor increases more as the discount multiplier increases.

3.4.5 An Application: Per-ride Tax

In this section, we illustrate how the proposed framework can be used to quantify the impact of system-level policy interventions. We consider a scenario where the government plans to impose a per-ride tax on only the ride-hailing (capacity 1) MoD service. This type of tax has already been considered in California [35].

Consider a government policy to charge the MoD service provider $2 on every ride-hailing trip. In the subsequent analysis, we assume that this cost is absorbed by the MoD provider with no increase to the passenger fare, but similar experiments where the cost is borne completely or partially by the traveler can also be considered. Using the discount factor function in scenario 1, we run the BO framework to optimize the supply-side parameters under this policy intervention. The optimal supply-side parameters, vehicle miles traveled (VMT) of MoD services, the ratio of the passenger miles traveled (PMT) and VMT, and the mode shares under tax and no-tax scenarios are shown in Table 3.8. Imposing the tax decreases the VMT of MoD services by 10.5% and decreases the profit by 30.5%.

In this tax scenario, running ride-hailing service is not as profitable as before. However, if the fleet size of the ride-hailing service is lowered, its demand is likely to shift to other travel modes. If the MoD service provider wants to induce this shift to move towards the other MoD modes and not public transit, the service provider needs to offer a higher discount for the shared services and thus the profit may further decrease. The optimal solutions of tax and no-tax scenario validate this hypothesis. A decrease in VMT of MoD services in tax scenario can also be attributed to a shift of demand from ride-hailing services to

Table 3.8: Comparison between the case with no tax and with tax.

	n_1	n_4	n_{10}	γ_4	γ_{10}	**Profit**
No tax	2826	235	526	0.12	0.24	145015
$2 tax	1388	1668	31	0.15	0.67	100748
	VMT	$\frac{\text{PMT}}{\text{VMT}}$	**Ride-hailing share (%)**	**Ridepooling share (%)**	**micro-transit share (%)**	**Public transit share (%)**
No tax	33835.7	1.11	61.0	13.5	19.2	6.3
$2 tax	30288.3	1.17	36.4	50.6	5.1	7.9

high capacity MoD services. In fact, the increase in the share of public transit by 1.6% also contributes to lower VMT of MoD services. The tax policy increases the revenue of public transit agency by $14340 which can further be recirculated for improvement in the level of service of the local public transportation to even further increase the transit ridership. One major benefit of the proposed framework is the ability to perform such scenario analysis and gain insights into the impacts of certain policy interventions.

CHAPTER 4

PRICING FOR MOBILITY-ON-DEMAND (MOD) SERVICES AT TRIP REQUEST LEVEL

4.1 Introduction

The popularity of ridesharing platforms such as Lyft and Uber has grown them into multi-million dollar markets [4]. These Mobility-on-Demand (MoD) services not only provide additional travel options to passengers but also have the potential to decrease auto ownership and pollution. Compared to the traditional taxi service, the MoD dispatch system can match the vehicles to trip requests in an online manner without a human dispatcher, which makes it possible to optimize the dispatch at scale [118]. Considering the high capacity on-demand fleet management problem, [1] proposed an anytime optimal algorithm to match the vehicle to trip requests with an objective to minimize the total delay (the waiting time and the detour caused by pooling) of the passengers. To leverage the information about future demand, [66] designed a proactive rebalancing method, which guided idle vehicles to future demand based on a probability distribution estimated using historical data. These studies focus on optimizing system-level metrics such as service rate and total delay without considering the interaction between the supply-side and the demand-side. In practice, the demand is endogenous in the system and passengers will make travel mode choice maximizing their utility given the trip metrics of each mode option [64]. This assumes an endogenous demand and optimizes the prices of each MoD service from the operator's profit-driven perspective.

Pricing is one of the most important tools to affect the interaction be-

tween the supply-side and the demand side in ridesharing systems, which has received considerable attention recently [4, 18, 42, 47, 54, 90, 118, 119]. Considering a crowdsourced ridesharing platform, [119] proposed a double auction-based pricing mechanism assuming individual rationality. Assuming both street-hailing and e-hailing services coexists in the system, [47] used a penalty success linear programming algorithm to optimize the pricing and penalty/compensation strategies to maximize the social welfare and maximize the revenue of taxi-hailing platforms, respectively. With knowledge of the traveler preference and the distribution of future trip requests, [90] formulated the MoD service profit maximization problem in a dynamic programming manner, and obtained a parametric rollout policy. However, most of these studies formulated the problem at the middle or macro level, and did not consider the real-application setting under some disputable assumptions. Vehicle dispatching and pricing problem are correlated, and the problem of finding the optimal pricing and vehicle assignment simultaneously for real-world MoD applications is still an open problem and one of the main research agendas in sharing economics [18].

In this chapter, we consider a multimodal transportation system, where three travel modes coexist – the exclusive MoD service (e.g., UberX), the sharing MoD service (e.g., Uberpool), and the taxi service. The above two MoD services are operated by one service operator. The objective of the work is to optimize the prices of both services for each trip request and the vehicle assignment to maximize the operator's profit. We assume that demand is endogenous in the system, and the travelers will make mode choice according to an estimated multinomial logit model [64, 90]. The main contributions of the work include:

- A sequential MoD service pricing framework, which can be integrated in the current ridesharing service (Uber, Lyft, etc) usage scenario. In the framework, whenever a service user requests a trip via the mobile phone application, we solve the pricing problem for each feasible vehicle around[1], and show the price corresponding to the assignment with the maximum expected profit to the user almost in real time. Given a vehicle and a trip request, we show that the pricing problem can be modeled as the multi-products pricing problem [45, 60], and provide the solution in closed-form in terms of Lambert W function.
- A batched MoD service pricing framework, which optimizes the prices of MoD services and vehicle assignment for the trip requests in each time interval[2]. In the framework, when a traveler requests a trip, the system does not the price of MoD services until the end of the corresponding time interval. It provides the service operator with an opportunity to leverage the information about all the requests coming in the same time interval, which is consistent with one of the state-of-the-art simulator [1]. For any two travelers that may share one ride using a given vehicle, we prove that the problem of optimizing prices of MoD services for both requests simultaneously is jointly concave if a certain condition about the cost parameter is satisfied, and we also show that the condition is usually satisfied in practice.

In summary, the proposed frameworks provide the service operator with tools to optimize the price and assignment in two different but realistic scenarios.

[1] A vehicle is feasible to the passenger if the vehicle can serve the trip request satisfying some predefined service level (wait time, travel time, etc).

[2] In our experiments, a time interval is 30 seconds.

4.2 Problem description and model formulation

We consider a monopolistic MoD service operator, which provides two types of service: exclusive service and sharing service. Specifically, each vehicle for the exclusive service can serve at most one traveler at any time, and each vehicle for the sharing service can serve several travelers simultaneously up to its capacity limit. Other than MoD services, the traditional taxi service also coexists in the system as a travel mode alternative. To maximize the profit of the MoD service operator, we want to determine: i) The prices for both the exclusive service and the sharing service for each trip (a vehicle and its assigned travel request(s)); ii) The assignment between the vehicle and the requests. The problem is formulated under the following assumptions:

Assumption 1. *Travelers and other transportation service providers (taxi service in our case) follow fixed strategies and will not react to different strategies of the operator.*

Assumption 2. *Both travelers and the MoD service operator have complete information on the price, travel time, and waiting time for a trip using the taxi service alternative for any origin-destination (O-D).*

Assumption 1 is a common assumption in the literature [90]. Assumption 2 is reasonable since the pricing formula for taxi service is usually publicly available.

In the following sections, we will present two pricing frameworks: i) A sequential MoD service pricing framework. In this framework, we show the prices for both services whenever travelers type in the origin and destination in the phone application; (2) A batching MoD service pricing framework, in which we display the prices to travelers after a short period (e.g., 30 seconds). In the

sequential pricing framework, since we have to show the price quickly, we have little time to optimize the assignment between the vehicle and the requests. As requests come in real-time, we sequentially conduct the pricing and assignment for each request. In the second framework, although travelers have to wait some time for the prices, the service operator can employ the time optimizing the vehicle assignment, and can optimize the prices for multiple travelers in a batch simultaneously. We will discuss both frameworks in details, but before that, we will first briefly discuss the mode choice model that will be used in both frameworks.

4.2.1 Mode choice model

We use a multinomial logit (MNL) model due to: i) its closed-form choice probability expression; and ii) its wide usage in the pricing literature [47, 60, 90]. We use the utility function form and coefficient estimates in the previous study [64]. The utility function for travel mode i and user j is as follows:

$$U_{i,j} = \beta_{p,j} \cdot p_{i,j} + \beta_{w,j} \cdot w_{i,j} + \beta_{t,j} \cdot t_{i,j} \tag{4.1}$$

where $U_{i,j}$ is the utility of user j choosing travel mode i, $p_{i,j}$, $w_{i,j}$, $t_{i,j}$ are the trip price, waiting time and travel time for user j choosing travel mode i, respectively, and $\beta_{p,j}$, $\beta_{w,j}$, and $\beta_{t,j}$ are the corresponding coefficients. The data used to estimate the model were obtained by conducting a stated-preference study in New York, in which the respondents were provided with a discrete choice experiment. The details of the experiment design can be found in the previous study [64]. In this work, we assume that the travelers in our system are homo-

geneous, but note that this assumption can be relaxed for other purposes in our framework. In the rest of the section, we omit the subscript for the traveler id.

Let the utility of the three travel modes (exclusive MoD service, sharing MoD service, taxi service) be U_e, U_s, U_t, respectively. The predicted probability of choosing travel mode i (P_i) is as follows:

$$P_i = \frac{e^{U_i}}{e^{U_e} + e^{U_s} + e^{U_t}} \tag{4.2}$$

4.2.2 Sequential MoD service pricing framework

In this section, we present the framework to conduct sequential pricing for MoD services. The specific procedure of the framework for a new request r is as follows:

i) Find feasible vehicles for both services for r, which is denoted as lists V_e and V_s, respectively;

ii) For each (v_e, v_s) where $v_e \in V_e$ and $v_s \in V_s$, we compute the optimal prices for both the exclusive service and the sharing service, and also record the expected profit of it;

iii) Show the traveler the prices corresponding to the maximum expected profit obtained in Step ii. If the traveler declines the trip service, the procedure is completed. Otherwise, go to Step iv;

iv) Show the traveler the assigned vehicle for the service the traveler chose. While the travel is waiting for their vehicle to arrive, the system reoptimizes the

vehicle assignment every a short period (e.g., 30 seconds).

We will first discuss how we determine the optimal price for a feasible trip (Step ii). Then, we will talk about all the other parts in details.

Pricing for a feasible trip

In this section, we consider the pricing problem for a feasible trip. Since we sequentially optimize the price for each coming request, we will always have two decision variables p_e and p_s, which are the price for the exclusive service and the sharing service, respectively. The service operator aims at maximizing the profit, which is the difference between the revenue and the operating cost (e.g. energy consumed, driver salary, etc). The operating cost for trips depends on travelers' mode choice decisions. Since the mode choice decision is not revealed until the prices are shown to the traveler, the service operator will maximize the expected profit $\underset{p_e,p_s}{\mathbb{E}}$ (profit), which is computed as follows:

$$\underset{p_e,p_s}{\mathbb{E}} \text{(profit)} = P_e \cdot (p_e - c) + P_s \cdot (p_s - m) \tag{4.3}$$

where c is the operating cost for the trip if the traveler chooses the exclusive service, and m is the expected operating cost for the trip if the traveler chooses the sharing service. Here m is an expectation because the cost depends on whether we will be able to share the traveler's ride with other travelers in the future and what will the cost be. Theoretically, the expectation cost for the given request should be computed as follows:

$$m = (1 - \sum_{r \in R_m} \alpha_r) \cdot c_s + \sum_{r \in R_m} \alpha_r (c_{r,m} - p_r) \tag{4.4}$$

where c_s is the cost if the request is not matched to any other requests in the future, R_m is the set of all possible future requests that might be matched to the given request, α_r is the probability that request r is matched to the given request conditional on that the given request accept the sharing trip, $c_{r,m}$ is the additional cost of adding request r to the matching with m, which is the difference between the cost of serving both requests and the cost of serving only the first request, p_r is the price we charge for request r. The distribution for each $r \in R_m$ cannot be computed in closed form, but these data can be obtained empirically from historical data.

In this work, we focus on the prices of MoD services, and regard everything else as constants. Given a vehicle, we can get the waiting time and travel time for the trip. Based on Assumption 2, we also know the trip metrics for the taxi service. Therefore, we can obtain the following equations:

$$U_e = \beta \cdot p_e + d_e \tag{4.5}$$

$$U_s = \beta \cdot p_s + d_s \tag{4.6}$$

$$e^{U_t} = D \tag{4.7}$$

where d_e, d_s and D are known constants, β is the coefficient for the price, which

is the same as β_p in Equation 4.1. Then, the objective function is in the following form:

$$\mathop{\mathbb{E}}_{p_e,p_s}(\text{profit}) = \frac{e^{\beta\cdot p_e+d_e}\cdot(p_e-c)+e^{\beta\cdot p_s+d_s}\cdot(p_s-m)}{e^{\beta\cdot p_e+d_e}+e^{\beta\cdot p_s+d_s}+D} \tag{4.8}$$

The objective function is in the same form as the multi-product pricing problem [45, 60]. Although the objective function is not concave in prices [45], researchers have shown that the problem is concave with respect to the market share, which is a one-to-one transformation of the price vector [60]. We show that for this specific case (two products), we can prove that there is one unique critical point and derive the closed-form of the solution in terms of the Lambert W function.

The partial derivative for p_s is:

$$\frac{\partial \mathop{\mathbb{E}}_{p_e,p_s}(\text{profit})}{\partial p_s} = \frac{\left(\beta\cdot e^{U_s}\cdot(p_s-m)+e^{U_s}\right)\cdot(e^{U_s}+e^{U_e}+D)-\beta\cdot e^{U_s}\cdot\left(e^{U_s}\cdot(p_s-m)+e^{U_e}\cdot(p_e-c)\right)}{(e^{U_s}+e^{U_e}+D)^2} \tag{4.9}$$

The partial derivative exists everywhere. Since critical points are always at the points where the partial derivatives do not exist or the partial derivatives are 0. Therefore, the critical points can only be at the points where the partial derivatives are 0. For simplicity, let e^{U_s} and e^{U_e} be x and y. Then, we can infer that $p_s = \frac{\log(x)-d_s}{\beta}$ and that $p_e = \frac{\log(y)-d_e}{\beta}$. A critical point will satisfy the following equations if it exists.

$$x + y + y \cdot \log(\frac{x}{y}) - (d_s + \beta \cdot m - d_e - \beta \cdot c) \cdot y + D \cdot \log(x) + D \cdot (1 - d_s - \beta \cdot m) = 0 \quad (4.10)$$

$$x + y - x \cdot \log(\frac{x}{y}) + (d_s + \beta \cdot m - d_e - \beta \cdot c) \cdot x + D \cdot \log(y) + D \cdot (1 - d_e - \beta \cdot c) = 0 \quad (4.11)$$

Let Equation 4.10 subtract Equation 4.11, and we can get the following conclusion:

$$x = e^{d_s + \beta \cdot m - d_e - \beta \cdot c} \cdot y \quad (4.12)$$

We can further infer that a critical point satisfies the following condition:

$$p_e - p_s = c - m \quad (4.13)$$

At a critical point, we can conclude that the difference between the price for the exclusive service and the sharing service is the expected cost savings if the traveler chooses the sharing service instead of the exclusive service. Employing this relationship in Equation 4.10, we can get the critical point:

$$p_e = \frac{\log(\frac{W\left((1+e^{d_s - d_e + \beta \cdot m - \beta \cdot c}) \cdot e^{d_e + \beta \cdot c - 1} / D\right) \cdot D)}{1 + e^{d_s - d_e + \beta \cdot m - \beta \cdot c}}) - d_e}{\beta} \quad (4.14)$$

As the Lambert W function is positive and increasing when the input is positive, the objective function 4.8 has a unique critical point. For a given user's origin and destination, we can check the boundary points and the critical point to find the price that gives the maximum profit.

Sequential pricing and fleet management in a network

In this section, we discuss all other steps mentioned at the beginning of Section 4.2.2, which are the procedure to find all feasible vehicles (Step i) and the procedure to reoptimize the vehicle assignment (Step iv).

We define that a vehicle is feasible for a trip request if it can serve the request under the following constraints:

- The waiting time must be below the maximum waiting time Ω.
- The total delay must not exceed the maximum delay Δ. The total delay is the difference between the actual arrival time and the earliest possible arrival time when a vehicle picks up the traveler immediately and follows the shortest path to the destination.
- For the vehicle, the number of passengers in the vehicle has to be smaller than or equal to the capacity at any time.

The constraints are consistent to the state-of-the-art fleet management approach [1], but note that the framework is agnostic of the *feasible* conditions. Whenever a new request comes in the system, we can check the location of all the vehicles and determine if they are feasible for the request. However,

since we would like to show prices to the traveler as soon as possible in the sequential pricing framework, we precompute and record the locations where an empty vehicle can serve the request satisfying the above conditions. Therefore, we only check the vehicles at such locations when a new request comes in the system.

If the traveler chooses one of the MoD services after we show him or her the optimal prices obtained in Step ii, we show the traveler the assigned vehicle and the expected pick-up time at once. While the traveler waits for the vehicle, we can leverage the information of the new coming requests and reoptimize the vehicle assignment. To not offend the traveler, we ensure that if we change the vehicle assignment, the new assignment will not increase the wait time and the total delay. We use a similar procedure to attain this goal as in the literature [1]. Specifically, the vehicle assignment is optimized every a short period (e.g., 30 seconds) by first finding all feasible trips and then use an integer program (IP) to choose the trips that optimize the objective. After completing the vehicle assignment, we also conduct the vehicle rebalancing based on the literature. Interested readers can refer to the previous study [1] for details. The difference in our procedure is that in our objective function, we consider the profit instead of the total delay.

4.2.3 Batching MoD service pricing framework

In this section, we present the batching MoD service pricing framework, where the prices for trip requests can be optimized in a batch. We assume that the capacity for the sharing service is 2 which is consistent with the current pool-

ing service such as Uberpool and Lyftline. The pricing method is integrated within the state-of-the-art fleet management approach [1] with major necessary changes. Specifically, the procedure is as follows:

i) Construct a pair-wise request-vehicle graph (RV-graph). RV-graph gives all possible pairwise matchings between vehicles and requests. In the graph, two requests r_i and r_j are connected if a virtual vehicle at the origin of either request can serve both under the constraints discussed in Section 4.2.2; Similarly, a request r is connected to a vehicle v if v can serve r without violating the constraints.

ii) Compute the Request-Trip-Vehicle graph (RTV-graph) using the cliques of the RV-graph. A feasible trip is defined as a set of requests that can be served by one vehicle without violating the constraints. RTV-graph consists of all the feasible trips and the vehicles that can serve them. A request r is connected to a trip T if T contains r; A trip T is connected to a vehicle v if v can serve T under the constraints. Step i and Step ii are the same as in the literature [1];

iii) Compute the Exclusive-Sharing-Vehicle (ESV) graph using both RV-graph and RTV-graph. For each request in the feasible trip in RTV-graph, we also connect it to each vehicle that can serve the request exclusively under the constraints according to RV-graph. We define that an Exclusive-Sharing-Vehicle (ESV) matching, which contains: a) a set of requests (2 in our case) that can share a ride in a vehicle based on RTV-graph; b) A vehicle that can serve the set of requests under the constraints; c) A vehicle for each request if the traveler chooses the exclusive service. For each ESV matching, we solve a pricing optimization problem to find the best prices to show to the travelers maximizing the expected profit. The details will be discussed in Section 4.2.3;

iv) Solve an ILP to find the optimal assignment from the vehicles to ESV matchings. The ILP is as follows:

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{M}} y_i \cdot u_i \\ \text{subject to} \quad & \sum_{i \in \mathcal{I}^{\mathcal{M}}_{V=j}} y_i \leq 1, \forall v_j \in \mathcal{V} \\ & \sum_{i \in \mathcal{I}^{\mathcal{M}}_{R=k}} y_i \leq 1, \forall k \in \mathcal{R} \\ & y_i \in \{0,1\}, \ \ \forall i \in \mathcal{M} \end{aligned} \tag{4.15}$$

where u_i is the expected profit of the ESV matching i. The decision variables are y_i where $y_i = 1$ if the ESV matching i is chosen. In the constraints, there are two sets: the set of ESV matchings that is connected to vehicle j is $\mathcal{I}^{\mathcal{M}}_{V=j}$; the set of ESV matchings that contain request k is $\mathcal{I}^{\mathcal{M}}_{R=k}$. The constraints ensure that: i) each vehicle will serve at most one ESV matching; ii) each request is either served or ignored.

Note that in the above formulation, we ensure that the request will always have the vehicle to serve it whatever the mode choice it chooses eventually (i.e., each vehicle can only be assigned to one ESV matching). In this case, some vehicles may be idling after travelers make their mode choice. As an alternative, we also propose a second IP formulation as follows, in which a vehicle can be assigned to multiple ESV matching. However, if a traveler chooses a MoD service and the assigned vehicle is assigned to other trip requests in an ESV matching, we impose a penalty c_p in the objective function.

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{M}} y_i \cdot u_i + \sum_{j \in \mathcal{V}} x_j & \\
\text{subject to} \quad & \sum_{i \in I^{\mathcal{M}}_{R=k}} y_i \leq 1, & \forall k \in \mathcal{R} \\
& y_i \in \{0,1\}, & \forall i \in \mathcal{M} \\
& x_j \leq 0, & \forall j \in \mathcal{V} \\
& x_j \leq (-c_p - \epsilon_j) \cdot (\sum_{i \in I^{\mathcal{M}}_{V=j}} y_i \cdot \gamma_{i,j} - 1), & \forall j \in \mathcal{V}
\end{aligned}
\tag{4.16}
$$

where $\gamma_{i,j}$ is the probability for the case that vehicle j is used after the ESV matching i is chosen. If vehicle j is for the exclusive service, $\gamma_{i,j}$ is the predicted probability of the corresponding traveler accepting the trip; If vehicle j is for the sharing service, $\gamma_{i,j}$ is the product of all the predicted probability of accepting the trip for the corresponding travelers in the sharing ride. In this model, one vehicle can be *overbooked* to multiple travelers. If more than one traveler accept the trip, then at least one of the traveler will be assigned to another vehicle or have no vehicle to serve. In this case, we denoted c_p as the penalty in dollars for each traveler that has no vehicle to serve, ϵ_j as the mean profit by vehicle j among all matchings it is feasible to, and x_j is the overbooking penalty for vehicle j. To formulate the number of travelers that have no vehicles assigned, there will be combinations of scenarios involved, and thus we use $\sum_{i \in I^{\mathcal{M}}_{V=j}} y_i \cdot \gamma_{i,j} - 1$ as an approximation.

v) After we show travelers prices and they make the mode choices, we change the maximum wait time and the maximum total delay for each request to the wait time and the total delay we display them so that the service level will not decrease for any traveler. Idling vehicles are rebalanced to the location of the unassigned requests assuming that they may request again in the future.

Pricing for a ESV matching

In this section, we show how to optimize the prices for the trip requests in an ESV matching. If the ESV matching contains only one request, the optimization procedure is the same as shown in Section 4.2.2. Therefore, we focus on the case where the ESV matching consists of two requests.

Suppose that r_1 and r_2 are the two requests in the ESV matching. The objective function will be in the following form given the vehicles in the ESV matching:

$$\begin{aligned} \mathop{\mathbb{E}}_{p_{1,e},p_{1,s},p_{2,e},p_{2,s}}(\text{profit}) = {} & P_{1,s} \cdot (p_{1,s} - c_{1,s}) + P_{2,s} \cdot (p_{2,s} - c_{2,s}) \\ & + P_{1,e} \cdot (p_{1,e} - c_{1,e}) + P_{2,e} \cdot (p_{2,e} - c_{2,e}) \\ & + P_{1,s} \cdot P_{2,s} \cdot (c_{1,s} + c_{2,s} - c_{s,s}) \end{aligned} \tag{4.17}$$

where $p_{1,s}$ and $p_{1,e}$ are the prices for the sharing service and the exclusive service for r_1, $p_{2,s}$ and $p_{2,e}$ are the prices for r_2, $c_{1,e}$ and $c_{2,e}$ are the operational cost for the company to serve r_1 and r_2 respectively using the exclusive service, $c_{s,s}$ is the operational cost for the company to serve the sharing trip of r_1 and r_2, and $c_{1,s}$ and $c_{2,s}$ are the expected operational cost for the company to serve r_1 and r_2 respectively if they do not form a sharing ride together. The cost is an expectation because it depends on whether the request will be shared with other requests and who it will be shared with in the future. The expected operational cost for each O-D is precomputed by running the simulation with an exogenous demand for a long period (e.g., a week). Specifically, we follow the procedure below:

i) As the demand is sparsely distributed on thousands of network nodes in a real-size network, we use the K-means clustering algorithm [71] to spatially cluster the nodes according to their geo-coordinates.[3]

ii) We run the simulation with an exogenous demand for a long period (e.g., a week), and record the operational cost for each trip based on the clusters where the origin and destination node belong to. After the simulation, we compute the mean of the recorded cost for each pair of clusters, and use it as the expected operational cost for each O-D pair that belongs to the cluster pair.

We now demonstrate that the objective function shown in Equation 4.17 is jointly concave in prices under some conditions. We first replace the price variables in the function by the following equations:

$$p_{1,s} = \frac{\log \frac{D_1 \cdot P_{1,s}}{1 - P_{1,s} - P_{1,e}} - d_{1,s}}{\beta} \tag{4.18}$$

where D_1 is exponential to the power of the utility of r_1 taking the taxi service. Similarly, we can also use the predicted probabilities to represent the other price variables. Since the price vector and the predicted vector has a one-to-one matching [60], we can obtain the price variables if we can find the optimal predicted probabilities. Therefore, minimizing the following function will be equivalent to maximizing the expected profit shown in Equation 4.17.

[3] Assuming that each traveler has a walking range of α miles, and we choose the number of clusters k to be the value satisfying $\frac{\text{Total area}}{k} \approx 2\pi\alpha^2$.

$$
\begin{aligned}
\beta \cdot \mathop{\mathbb{E}}_{P_{1,e},P_{1,s},P_{2,e},P_{2,s}}(\text{profit}) = {} & P_{1,s} \cdot (\log(\frac{D_1 \cdot P_{1,s}}{1 - P_{1,s} - P_{1,e}}) - \beta \cdot c_{1,s}) \\
& + P_{1,e} \cdot (\log(\frac{D_1 \cdot P_{1,e}}{1 - P_{1,s} - P_{1,e}}) - \beta \cdot c_{1,e}) \\
& + P_{2,s} \cdot (\log(\frac{D_2 \cdot P_{2,s}}{1 - P_{2,s} - P_{2,e}}) - \beta \cdot c_{2,s}) \\
& + P_{2,e} \cdot (\log(\frac{D_2 \cdot P_{2,e}}{1 - P_{2,s} - P_{2,e}}) - \beta \cdot c_{2,e}) \\
& + P_{1,s} \cdot P_{2,s} \cdot \beta \cdot (c_{1,s} + c_{2,s} - c_{s,s})
\end{aligned} \tag{4.19}
$$

We can derive the Hessian matrix of the above equation H as follows, where the order of each row and column is $P_{1,s}$, $P_{1,e}$, $P_{2,s}$ and $P_{2,e}$:

$$
\begin{bmatrix}
\frac{1}{P_{1,s}} + \frac{1}{1-P_{1,s}-P_{1,e}} + \frac{1}{(1-P_{1,s}-P_{1,e})^2} & \frac{1}{1-P_{1,s}-P_{1,e}} + \frac{1}{(1-P_{1,s}-P_{1,e})^2} & \beta \cdot (c_{1,s} + c_{2,s} - c_{s,s}) & 0 \\
\frac{1}{1-P_{1,s}-P_{1,e}} + \frac{1}{(1-P_{1,s}-P_{1,e})^2} & \frac{1}{P_{1,e}} + \frac{1}{1-P_{1,s}-P_{1,e}} + \frac{1}{(1-P_{1,s}-P_{1,e})^2} & 0 & 0 \\
\beta \cdot (c_{1,s} + c_{2,s} - c_{s,s}) & 0 & \frac{1}{P_{2,s}} + \frac{1}{1-P_{2,s}-P_{2,e}} + \frac{1}{(1-P_{2,s}-P_{2,e})^2} & \frac{1}{1-P_{2,s}-P_{2,e}} + \frac{1}{(1-P_{2,s}-P_{2,e})^2} \\
0 & 0 & \frac{1}{1-P_{2,s}-P_{2,e}} + \frac{1}{(1-P_{2,s}-P_{2,e})^2} & \frac{1}{P_{2,e}} + \frac{1}{1-P_{2,s}-P_{2,e}} + \frac{1}{(1-P_{2,s}-P_{2,e})^2}
\end{bmatrix}
$$

If Equation 4.19 is jointly convex in the predicted probabilities, we know that the problem has at most one critical point, and we can find the optimal solution by finding the point satisfying the first-order derivative condition and compare its objective function value with the boundary points. In other words, if we can use the second-order derivative test to show that the Hessian matrix is positive definite, the problem is solved.

Let $\vec{y} = y_1, y_2, y_3, y_4$ be a non-zero vector of four any real numbers. We can derive the following:

$$\begin{aligned}\vec{y}\cdot H\cdot\vec{y}^T &= (\frac{1}{1-P_{1,s}-P_{1,e}}+\frac{1}{(1-P_{1,s}-P_{1,e})^2})\cdot(y_1+y_2)^2\\ &+(\frac{1}{1-P_{2,s}-P_{2,e}}+\frac{1}{(1-P_{2,s}-P_{2,e})^2})\cdot(y_3+y_4)^2\\ &+\frac{y_1^2}{P_{1,s}}+\frac{y_2^2}{P_{1,e}}+\frac{y_3^2}{P_{2,s}}+\frac{y_4^2}{P_{2,e}}+2\cdot y_1\cdot y_3\cdot\beta\cdot(c_{1,s}+c_{2,s}-c_{s,s})\end{aligned} \tag{4.20}$$

If $\frac{y_1^2}{P_{1,s}}+\frac{y_3^2}{P_{2,s}}+2\cdot y_1\cdot y_3\cdot\beta\cdot(c_{1,s}+c_{2,s}-c_{s,s})\geq 0$, $\vec{y}\cdot H\cdot\vec{y}^T\geq 0$ because all the other parts in the equation are non-negative. Let $C=c_{1,s}+c_{2,s}-c_{s,s}$ for readability, we can derive the following:

$$\frac{y_1^2}{P_{1,s}}+\frac{y_3^2}{P_{2,s}}+2\cdot y_1\cdot y_3\cdot\beta\cdot C=(\sqrt{-\beta\cdot C}\cdot(y_1-y_3))^2+(\frac{1}{P_{1,s}}+\beta\cdot C)\cdot y_1^2+(\frac{1}{P_{2,s}}+\beta\cdot C)\cdot y_3^2 \tag{4.21}$$

Therefore, the condition to prove the problem is jointly concave in the predicted probabilities is the following:

$$C\leq\min\{-\frac{1}{\beta\cdot P_{1,s}},-\frac{1}{\beta\cdot P_{2,s}}\} \tag{4.22}$$

Since $\beta<0$, $P_{1,s}\leq 1$ and $P_{2,s}\leq 1$, the following condition is a sufficient condition to satisfy Equation 4.22:

$$C\leq-\frac{1}{\beta} \tag{4.23}$$

where $C=c_{1,s}+c_{2,s}-c_{s,s}$ is the operational cost savings if r_1 and r_2 can share a ride compared to the sum of the expected cost for either of them choosing the sharing service. The cost savings should always be larger than 0. Otherwise,

the service provider should not match r_1 and r_2 as a sharing trip. We know that $c_{s,s} \geq \max\{c_{1,e}, c_{2,e}\}$, $c_{1,s} \leq c_{1,e}$ and $c_{2,e} \leq c_{2,e}$, and thus Equation 4.23 is satisfied if the following equation is satisfied:

$$\min\{c_{1,e}, c_{2,e}\} \leq -\frac{1}{\beta} \tag{4.24}$$

According to our estimated mode choice model discussed in Section 4.2.1, $-\frac{1}{\beta} \approx 13.5$. In other words, if the operational cost of the shorter trip has a cost smaller than \$13.5, the problem is jointly concave in the predicted probabilities, and we can compute the optimal price values. We will use taxi trips in Manhattan, New York City (NYC) in numerical experiments. Manhattan Island is 22.7 square miles, 13.4 miles long and 2.3 miles wide at its widest. The driving cost is about \$0.1458 per mile including the cost of fuel, maintenance and tiers [64] using a self-driving fleet. Therefore, if the shorter trip among r_1 and r_2 is shorter than 92.6 miles, the problem is jointly concave in the predicted probabilities, which should be satisfied in the MoD context. If the fleet is operating by human drivers and the vehicles are owned by the drivers, the company will charge a fixed proportion of all fairs as the commission fee, and there will be no per mile cost imposed on the company. Assuming that drivers will obey the dispatch order sent by the company, the model is still applicable.

If the condition is not satisfied in practice for some ESV matchings, we can use a brute-force based method to find the solution. Specifically, we can enumerate $P_{1,s}$ in the range $[0, 1]$ with a predefined step length. Given the value of $P_{1,s}$, we can prove that the problem to optimize $P_{1,e}$, $P_{2,s}$ and $P_{2,e}$ is jointly concave in the variables using similar methods. After enumerating all the candidate $P_{1,s}$, we can return the prices with the highest expected profit.

CHAPTER 5

STOCHASTIC ON-TIME ARRIVAL PROBLEM IN TRANSIT NETWORKS

5.1 Introduction

A vast majority of previous studies on routing problems in stochastic transportation networks aim to identify the route or adaptive policy with the least expected travel time (LET) [17, 33, 39, 44, 51, 68, 76, 89, 113, 115]. While the expected travel time is a natural optimality metric, there exists a variety of situations where it is not sufficient, and tail statistics must be considered; for instance, travelers who want to catch a flight are more concerned with arriving on time with a high probability rather than with minimizing their expected travel time [116]. To account for these contexts, other formulations that consider a reliable optimal path have been proposed, starting with [38]. In this formulation, the goal is to find the path that maximizes the probability of realizing a travel time smaller than some desired time budget. This definition is subsequently extended to the online context in [34], and is commonly referred to as the Stochastic On-Time Arrival (SOTA) problem. In the SOTA problem, the weight of each network link is a random variable with a known probability density function that represents the travel time of the link. The solution to the SOTA problem is a routing policy that maximizes the probability of arriving at the destination within a specified time budget. Here the routing policy is an adaptive solution that determines the routing decision at each node based on the realization of the travel time experienced en-route up to that point.

Given the complexity of the SOTA problem, significant efforts have been made to design efficient solution algorithms. [32] formulate it as a dynamic pro-

gramming problem and use a standard successive approximation (SA) procedure. This approach, however, has no finite bound on the maximum number of iterations needed for convergence in networks with loops. In [95,96], the authors propose a label-setting algorithm to resolve this issue, exploiting the fact that there is always a non-zero minimum realizable travel time on each link in road networks. To further reduce the computation time, [94] modify two deterministic shortest path preprocessing techniques: reach [43] and arc-flags [7,49], and apply them to the SOTA problem. Other studies [82–84,87] explore computationally efficient solution strategies for providing reliability guarantees in stochastic shortest path problems, where a fixed route (as opposed to a policy) is desired.

In transit networks, the problem of determining a set of reliable travel decisions is significantly more complex than in road networks. In this context, the traveler is not in control of the transit line(s) that they may travel on to reach the destination, and may have to make a number of complex decisions regarding which transit line(s) to take. This complexity arises from the following characteristics of transit systems.

- **Uncertainty of arrival times/headways:** While the uncertainty in road networks is limited to the travel time, in transit networks one also needs to consider the uncertainty of arrival times and headways. Whether to take a transit line that arrives at a station or wait for a potentially faster service that is yet to arrive depends on the probabilistic trade-off between the extra waiting time and potential travel time savings. Therefore, computing the solution also requires modeling and solving for the headway distributions.

- **Combinatorial choice set:** In road networks, it never pays off to idle at a node and delay the departure from the node in hopes of improving the probability of arriving at the destination on time. However, in transit networks, it may be advantageous to not take the first transit line that arrives at the station (that can get you to your destination) and wait for a better option (e.g. wait for an express train or a more direct bus line). Therefore, a passenger needs to choose between multiple transit lines sharing segments of routes, and the decision regarding which transit line to board at a station depends on the unknown future arrival order of the candidate transit lines, which is an exponentially large set in the number of candidate (feasible) lines at each station, and leads to a combinatorial choice set.

Previous research on transit routing problems typically simplifies the problem by assuming that passengers board the first arriving transit service from an attractive line set that is precomputed to minimize the expected total travel time [21, 62, 85, 104]. Under this assumption, the passenger is committed to a transit line set. However, the traveler does not need to make a boarding decision until a transit vehicle is about to depart ([48]), and it can be meaningful to adapt decisions based on information learned (uncertainties that are realized) during the trip. Fortunately, the role of online information has been noted recently in multiple studies. [41] propose a frequency-based assignment model under the assumption that the arrival time of the next transit vehicle is available once the passenger arrives at a transit station and demonstrate the potential benefits of utilizing such information. Instead of assuming full information, [19] propose a routing strategy in a transit network with partial online information at the stations. The partial online information represents that the arrival time of the

incoming transit vehicles is available only for a subset of the candidate transit lines. [86] develop a transit assignment model which considers two types of available information (partial information and full information), and demonstrate the impact of online information on assignment results. In our work, we consider the routing problem in transit networks as a fully online problem. The online information setting mentioned above can be easily adopted into our framework by changing the travel time and headway distributions accordingly.

The aim of this research is to formulate the SOTA problem for transit networks and develop efficient algorithms to solve it in this setting. The specific contributions of the work include:

- The formulation of the SOTA problem for transit networks, including a general network structure for stochastic transit networks and a decision-making model. Both the waiting time for each transit line and the travel time on each link are assumed to be random variables with known probability density functions. The solution is an adaptive policy that fully considers the transit specific characteristics of the problem mentioned above.
- The design of a dynamic programming (DP) algorithm, which is pseudo-polynomial in the number of transit stations and time budget, and exponential in the number of transit lines at each station, which is practically a small number. To reduce the search space, we develop a definition of transit line dominance and present methods to identify this transit line dominance, which significantly decreases the computation time in our numerical experiments.
- Extensive experiments in a synthetic network and in the Chicago transit network, which show the potential for solving this problem in a real-time

route planning application setting. We also propose a general procedure to generate travel time, headway, waiting time distributions from General Transit Feed Specification (GTFS) data.

The rest of this chapter is organized as follows. In Section 5.2, we formulate the mathematical problem, in particular, we describe the network model and the underlying decision-making framework. In Section 5.3, we introduce a dynamic programming based approach to solve the problem, as well as the complexity analysis of the algorithm. In Section 5.4, we provide a series of algorithmic techniques to reduce the search space. Section 5.5 consists of numerical results on the computational performance and practical efficiency of the model and algorithms introduced in this work, both in a synthetic network and in the Chicago transit network.

5.2 Problem formulation.

5.2.1 Problem description

In this section, we extend the SOTA problem definition to the context of transit networks. In this setting, as mentioned previously, two types of random variables need to be considered: the travel time between two transit stations and the waiting time for each transit line arrival at each transit station. The objective of the SOTA problem for transit networks is to find the routing (boarding, alighting and transferring) policy that maximizes the probability of arriving at the destination within a time budget. For conciseness, in the remainder of the article,

we use *utility* to represent *the probability of arriving at the destination within the remaining time budget*, but note that the framework can be extended to other functions, and to robust settings [37]. The routing policy here includes the boarding, alighting and transferring decisions at each transit station in different situations (time already spent waiting at a station, arrival order of transit services, etc).

The following modeling assumptions are made.

Assumption 3. *Passengers' arrivals are independent of the transit schedule.*

This assumption models a situation where the transit service is frequency-based (not schedule-based) and passengers do not adjust their specific departure times based on the transit schedule. This is a standard assumption made in the literature [41, 62]. In modern services, transit vehicles' positions may be published in real-time (even for frequency-based services), and thus passenger arrivals may be correlated with the bus schedule. The formulation can be adapted to account for this by changing the travel time distribution and waiting time distribution accordingly.

Assumption 4. *Only the first arrival from each transit line (after the passenger enters the station) is considered as a candidate in the choice set.*

This assumption is made for both modeling and algorithmic reasons:

- Guiding the passenger to ignore the first arrival from a particular line, but board the second arrival from that same transit line makes the routing direction confusing;

- Without this assumption, it is possible for the passenger to have to make an infinite number of decisions in the theoretical worst case (albeit with vanishing probability).

The passenger may not be able to board certain transit services (especially during rush hour or after a major event), if the transit service is full and has no remaining capacity. However, since we do not have data about the occupancy of each transit vehicle, the capacity of each transit service is not considered in our study. Note that this is not a disadvantage of our framework because given the data for the occupancy of transit vehicles, we should be able to define the waiting time to be the waiting time for the first arrival transit service that the passenger can board, and our framework can then be adapted to use these distributions without any changes.

Assumption 5. *No two transit services can arrive at the transit station at the same time.*

Without loss of generality, two events never occur at exactly the same time in the continuous setting. In the subsequent discretized form of the problem, this assumption translates to no two transit services arriving at a transit station during the same discretized time interval[1].

5.2.2 Transit network representation

We consider a directed graph $G(V, E)$, in which V is the set of nodes and E is the set of links. We model the transit stations with three types of nodes:

[1]While the optimality of the solution relies on this assumption, if two lines arrive in the same time interval in practice, the algorithm can closely approximate the solution by considering the two cases of one arriving before the other.

- **Station nodes:** A station node denoted by S_y^X represents a passenger waiting at station y for the set of candidate transit lines X.
- **Arrival nodes:** An arrival node denoted by $A_y^{i,X}$ represents transit line i arriving first at station y among all the candidate lines in $X \cup \{i\}$, and transit lines $j \in X$ having not arrived yet, since the passenger arrives at the station.
- **Line nodes:** A line node denoted by L_y^i represents the passenger boarding transit line i at station y.

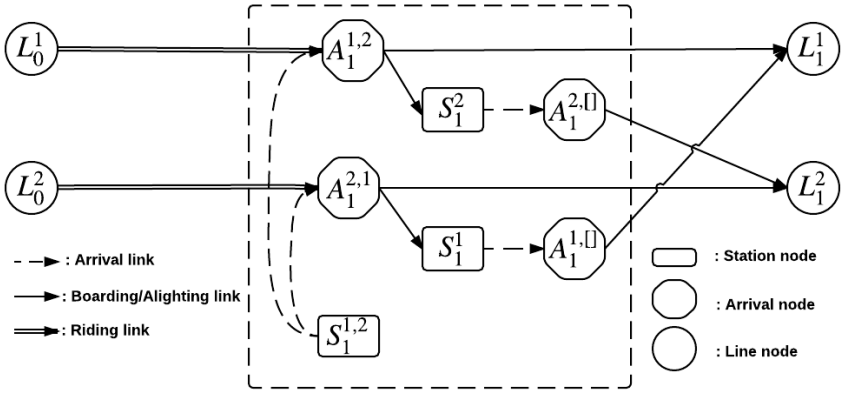

Figure 5.1: Network representation of a transit station served by two transit lines. All the links and nodes within the dashed line rectangle are used to model the decision-making at a single physical station.

The set of station nodes, arrival nodes and line nodes are V_S, V_A and V_L respectively. Without loss of generality, the OD is selected from the station nodes to model a passenger starting their trip from a station node in the waiting state. Give a passenger in a waiting state at a station node with m candidate transit lines, it is possible for any of the m lines to arrive first. In each of these situations, the passenger either boards the transit line that arrives or continues waiting for other candidate lines. Passengers make decisions that maximize their utility.

We categorize links as follows:

- **Arrival links:** Links from station nodes to arrival nodes.
- **Riding links:** Links from line nodes to arrival nodes.
- **Boarding links:** Links from arrival nodes to line nodes.
- **Alighting links:** Links from arrival nodes to station nodes.

Figure 5.1 illustrates the network representation of the decision-making process at a transit station with two transit lines. A passenger starting the trip physically at station 1 starts the trip at station node $S_1^{1,2}$ in the model. If transit line 1 arrives first, the passenger moves to the corresponding arrival node $A_1^{1,2}$. The passenger has to decide to either board this transit line or continue to wait for line 2. If the passenger continues waiting, the passenger moves to station node S_1^2. Otherwise, the passenger moves to line node L_1^1. All the links and nodes within the dashed line rectangle are used to model the decision-making at a single physical station. This network model allows for transfers. For instance, a passenger having boarded line 1 at station S_0 (the station preceding station S_1) is able to get to station node S_1^2 and wait for the next arrival of line 2.

The link costs in the network are defined as follows. A riding link (i, j) is associated with a travel time distribution $p_{i,j}(\cdot)$. A boarding link has no cost, since it refers to an instantaneous event. An arrival link is associated with a waiting time distribution $w_y^j(\theta, r)$, characterizing the probability density for the waiting time being θ for transit line j at station y, given that the passenger has already waited at the station for r units of time. Note that θ is the waiting time on top of r, and models transit line j arriving at station y after a total waiting time of $(r + \theta)$ since the passenger arrives at the station. Therefore, $w_y^j(\theta, 0)$ is the original

waiting time distribution when the passenger arrives at the station. Research has shown that empirical headway data fit better with Loglogistic, Gamma and Erlang distributions than the exponential distribution [62]. Therefore, we do not assume that the waiting time distribution is memoryless (as some other studies do), and use the following rule to normalize the waiting time distribution given r:

$$w_y^j(\theta, r) = \frac{w_y^j(r+\theta, 0)}{1 - \int_0^r w_y^j(\alpha, 0) d\alpha} \quad 0 \le \theta \le T - r, 0 < r \le T \tag{5.1}$$

with T being the total time budget when the passenger begins the trip.

In Section 5.2.3, we discretize the time-space to solve the problem numerically. In the discretized space, we force the waiting time to be at least one unit of the discretized time interval. Therefore, the waiting time equation reads as follows.

$$w_y^j(\theta, r) = \frac{w_y^j(r+\theta, 0)}{1 - \sum_0^r w_y^j(\alpha, 0) d\alpha} \quad 1 \le \theta \le T - r, 0 < r \le T \tag{5.2}$$

Although the waiting time distribution is a 2-D array, as explained in Section 3.2, we can save computation time in practice by precomputing and storing $1 - \int_0^r w_y^j(\alpha, 0) d\alpha$ for each r after we discretize the time-space.

5.2.3 The SOTA problem for transit networks

In this section, we describe how to compute the utility functions of the passenger at the different types of nodes in the model. The utility at a node i is a

function of the remaining time budget t when the passenger arrives at the node, denoted by $u_i(t)$. The utility function at the destination node D is:

$$u_D(t) = 1 \quad 0 \leq t \leq T$$

since the passenger has completed the trip when at node D.

Line nodes

Recall from Figure 5.1 that a line node only contains an outgoing edge to an arrival node. Assume that the line node we are considering is i and the following arrival node is j. The travel time on link (i, j) is a random variable θ, and the remaining time budget at node j is $t - \theta$. Therefore, the utility at line node i is a function of the remaining time budget t when the passenger arrives at the node, denoted by $u_i(t)$.

Definition 1. *The utility function at a line node i can be computed as follows.*

$$u_i(t) = \mathop{\mathbb{E}}_{\theta}(u_j(t-\theta)) = \int_0^t p_{i,j}(\theta) \cdot u_j(t-\theta)d\theta, \forall i \in V_L, (i,j) \in E, 0 \leq t \leq T$$

where $j \in V_A$ is the subsequent arrival node of line node i.

This utility function is analogous to the standard utility function of the SOTA problem for road networks.

Arrival nodes

All the routing decisions occur at arrival nodes, since the passenger is faced with the decision of either boarding the transit line that arrives first (selecting the subsequent line node) or continuing to wait for the other candidate transit lines (selecting the corresponding station node). Let $A_y^{i,X}$ be the arrival node of interest, and $u_{A_y^{i,X}}(t, r)$ be the utility when the passenger has a remaining time budget t and has already waited at the station for r units of time. We call the tuple (t, r) the passenger state. Since the passengers aim to maximize the utility, the utility at an arrival node is the maximum of the utilities at the subsequent line node and station nodes.

Definition 2. *The utility function at an arrival node $A_y^{i,X}$ with passenger state (t, r) is:*

$$u_{A_y^{i,X}}(t, r) = \max_{j \in V_L | (A_y^{i,X}, j) \in E;\ S_y^X \in V_S | (A_y^{i,X}, S_y^X) \in E} \{u_j(t), u_{S_y^X}(t, r)\}$$

Station nodes

Station nodes are followed only by arrival nodes, and the utility at a station node is the expectation of the utility at the following arrival nodes. An arrival node is defined by the first arriving transit line and the corresponding passenger state. Let (i, θ) be a random event which represents that transit line i is the first arriving transit line, and the waiting time for it is θ. Assume that S_y^X is the station node of interest, and that we want to compute $u_{S_y^X}(t, r)$. For each $0 \leq \theta \leq t$, the probability of the transit line i arriving the first after waiting θ units of time is $w_y^i(\theta, r) \cdot \prod_{j \in X \setminus i} (1 - \int_0^\theta w_y^j(\alpha, r)\, d\alpha)$. In this case, the passenger has waited $\theta + r$ units of time in total, and the utility at this arrival node is $u_{A_y^{i,X\setminus i}}(t - \theta, \theta + r)$.

Definition 3. *The utility function at a station node is:*

$$
\begin{aligned}
u_{S_y^X}(t,r) &= \mathop{\mathbb{E}}_{(i,\theta)}\left(u_{A_y^{i,X\setminus i}}(t-\theta,\theta+r)\right) \\
&= \sum_{i\in X}\left(\int_0^t w_y^i(\theta,r)\cdot\prod_{j\in X\setminus i}(1-\int_0^\theta w_y^j(\alpha,r)d\alpha)\cdot u_{A_y^{i,X\setminus i}}(t-\theta,\theta+r)d\theta\right)
\end{aligned}
$$

This utility is the weighted average of the utility at the corresponding arrival nodes.

The integrals in the above equations do not have closed form expressions and cannot be solved analytically. Therefore, they need to be integrated numerically by discretizing the time horizon into small intervals. In the discretized model, we assume that no two transit lines can arrive at the same transit station at the same discretized time interval. As a consequence, the solution not guaranteed to be optimal, since there is a small (non-zero) probability that two lines might arrive at the same time interval regardless of how small the discretization is. However, we can set the length of time intervals to be a small number in practice[2].

Given a fixed time discretization, the discrete form of the utility function at station node reads as follows.

Definition 4. *The discrete form of the utility function at station nodes is:*

$$
u_{S_y^X}(t,r) = \sum_{i\in X}\left(\sum_{\theta=1}^{t} w_y^i(\theta,r)\prod_{j\in X\setminus i}(1-\sum_{\alpha=0}^{\theta} w_y^j(\alpha,r))\cdot u_{A_y^{i,X\setminus i}}(t-\theta,\theta+r)\right)
$$

[2]In the experiments, we set the time interval length to be 15 seconds. Using this time interval length, the average probability of multiple buses arriving at the same time interval for each station is about 0.6% in the Chicago transit network we use in numerical experiments. For a particular OD, this probability is usually lower since the candidate bus line set is a subset.

.

The utility function for other types of nodes can be discretized similarly.

Example 1. *Here we show a simple example that shows the sophisticated decision-making process that the model allows. Assume that the passenger is waiting at a station with three candidate transit lines. Table 5.1 is the waiting time distribution for each transit line, and the utility of taking the transit line corresponding to the waiting time. Figure 5.2 shows the optimal decision based on the arrival order. We can observe that: (1) The transit line that first arrives is not always the optimal choice. When transit line 3 arrives at time 2, it is the first transit service that arrives. However, it is optimal for the passenger to not board, and continue to wait; (2) Even when it is optimal to not board a transit line at an arrival event, the event may impact the future optimal decisions, as illustrated by the differing policies following the arrival or non-arrival of transit line 3 at time 2.*

Table 5.1: The waiting time and utility distribution for example 1.

Transit line ID	Waiting time	Probability	Utility
1	1	0.05	0.90
	3	0.05	0.80
	10	0.90	0.00
2	5	0.90	0.85
	15	0.10	0.00
3	2	0.50	0.70
	6	0.50	0.60

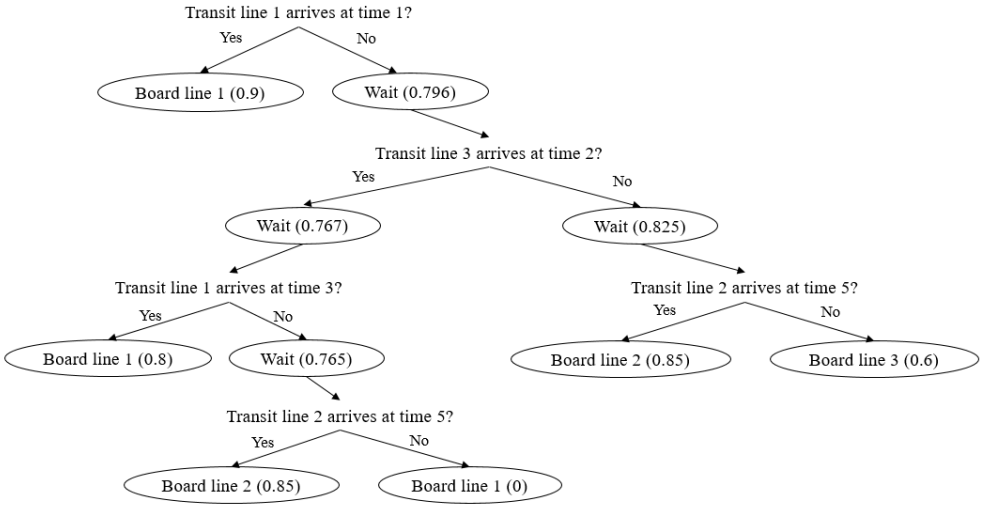

Figure 5.2: The optimal boarding decision corresponding to the transit line arriving order for example 1. The number in each decision node (board and wait) represents the decision's utility. Note that arrival events for which boarding is never optimal are ignored in the figure (e.g. the arrival of transit line 1 at time 3 following the non-arrival of transit line 3 at time 2).

5.3 Solving the SOTA problem for transit networks

In this section, we describe how to solve the discrete time version of the SOTA problem for transit networks. We first show how the problem can still (even in the transit setting) be solved using a dynamic programming approach, then present a complexity analysis of the algorithm, and finally in Section 5.4 describe a number of search space pruning methods designed to make the problem tractable in practice.

5.3.1 Dynamic programming approach

In [96], a label-setting algorithm for the SOTA problem is developed by exploiting the fact that each link in a road network has a positive minimum realizable travel time. This fact guarantees that there will be no loops in the network with zero travel time, and thus allows for a dynamic programming approach for solving the problem. However, in the transit network representation proposed in the previous section, there are links with zero minimum realizable travel time.

Claim 4. *In the transit network representation introduced in Section 5.2, there is no loop with a zero realizable travel time.*

Proof. Only the links starting from the arrival nodes can have zero minimum realizable time. To form a loop, the passenger has to first get to an arrival node from some other type of node, which is not an arrival node. The links originating from all other types of nodes have a positive minimum realizable travel time, so a zero travel time loop is not possible. ■

Therefore, we can still use a dynamic programming approach to solve the SOTA problem for transit networks. Each sub-problem in the dynamic program can be defined by (i, t, r) where i is the node that we consider, t and r have the same definition as in Section 2. Let $\text{OPT}(i, t, r)$ denote the utility at node i when the passenger has a time budget of t and has waited at the station for r units of time. Note that the $\text{OPT}(i, t, r)$ satisfies the Bellman's Principle of Optimality, i.e., the remaining decisions only depend on the current state and are not related to the past states and decisions. According to the definitions in Section 2, we claim that $\text{OPT}(i, t, r)$ satisfies the following recurrence relation:

$$\mathrm{OPT}(i,t,r) = \begin{cases} 1 & \text{if } i \text{ is the destination} \\ 0 & \text{if } t < 0 \\ \sum_{0}^{t} p_{i,j}(\theta) \cdot \mathrm{OPT}(j, t-\theta, 0) & \text{if } i \text{ is a line node and } (i,j) \in E \\ \max_{j \in V_L | (A_y^{i,X}, j) \in E;\ S_y^X \in V_S | (A_y^{i,X}, S_y^X) \in E} \{\mathrm{OPT}(j,t,0), \mathrm{OPT}(S_y^X, t, r)\} & \text{if } i \text{ is arrival node } A_y^{i,X} \\ \sum_{i \in X} \left(\sum_{\theta=1}^{t} w_y^i(\theta, r) \prod_{j \in X \setminus i} (1 - \sum_{\alpha=0}^{\theta} w_y^j(\alpha, r)) \cdot \mathrm{OPT}(A_y^{i,X\setminus i}, t-\theta, \theta+r) \right) & \text{if } i \text{ is station node } S_y^X \end{cases} \tag{5.3}$$

The initial problem we wish to solve is given by $(O, T, 0)$, i.e., the passenger is at the origin node O with time budget T and has waited for zero units of time. The algorithm first initializes the utility function corresponding to the destination node to 1 and computes all the normalized waiting time distributions that are needed in the subsequent computation. Then, the utility at each node is updated in a dynamic programming fashion according to the recurrence relation given in Equation 5.3.

5.3.2 Complexity analysis

The runtime of the algorithm is pseudo-polynomial in the number of stations in the transit network and time budget, and exponential in the number of transit lines at any station. We first provide the number of nodes of all three types. Then, we analyze the time complexity for computing the utility on each type of node.

Claim 5. *For a station with m transit lines, there are i) m line nodes, ii)* $\sum_{i=1}^{m} C_m^i$ *station nodes, and iii)* $\sum_{i=1}^{m} i \cdot C_m^i$ *arrival nodes.*

Proof. i) For every transit line, there is a corresponding line node. Therefore, there are m line nodes in total. ii) For each combination of transit lines representing the non-empty set of lines yet to arrive, there is a station node. Therefore, there are $\sum_{i=1}^{m} C_m^i$ station nodes in total. iii) For each station node, any transit line can be the line arriving the first and yield an associated arrival node. Therefore, there are $\sum_{i=1}^{m} i \cdot C_m^i$ arrival nodes. ■

We now analyze the time complexity of the algorithm for a transit network with a station set Y and each station has no more than m transit lines. Let M_y be the set of transit lines at station y. The algorithm first computes all the normalized waiting time distributions that are needed in the subsequent computation, which takes $O(|Y| \cdot m \cdot T^3)$ time based on Equation 5.1. In our implementation, we use $O(|Y| \cdot m \cdot T)$ memory to store $1 - \sum_{\alpha=0}^{\theta} w_y^j(\alpha, 0), \forall \theta \leq T, y \in Y, j \in M_y$, which decreases the time complexity to $O(|Y| \cdot m \cdot T^2)$.

Claim 6. *For a station set Y in which each station has no more than m transit lines, the time complexity of computing the utility functions for a time budget of T is:*

- $O(|Y| \cdot m \cdot 2^{m-1} \cdot T^2)$ *for all arrival nodes,*
- $O(|Y| \cdot (m^2 - m) \cdot 2^{m-2} \cdot T^3)$ *for all station nodes.*

Proof. Based on Definition 2, we need to update the utility functions at each arrival node for each possible remaining time budget t and each waiting time r. In addition, based on Claim 5, there are $\sum_{i=1}^{m} i \cdot C_m^i$ arrival nodes for a station with m transit lines, so it takes $O(|Y| \cdot \sum_{i=1}^{m} i \cdot C_m^i \cdot T^2) = O(|Y| \cdot m \cdot 2^{m-1} \cdot T^2)$ time to compute the utility at all the arrival nodes.

Assume that the station node of interest is S_y^X. To compute $U_{S_y^x}(t, r)$, for each

transit line $i \in X$, we need to compute the probability of line i arriving among all lines in X, which is $\sum_{\theta=1}^{t} w_y^i(\theta, r) \prod_{j \in X \setminus i}(1 - \sum_{\alpha=0}^{\theta} w_y^j(\alpha, r))$ according to Definition 4. Therefore, for each $i \in X$ and passenger state (t, r), we need $O((|X| - 1) \cdot T^2)$ to compute the above probabilities. Based on Definition 4, we need to update the utility functions at each station node for each possible remaining time budget t and each waiting time r. Consequently, we need $O(|X| \cdot (|X| - 1) \cdot T^4)$ to update the utility functions for each station node in total. According to Claim 5, there are $\sum_{i=1}^{m} C_m^i$ station nodes for a station with m transit lines. Therefore, it takes $O(|Y| \cdot \sum_{i=1}^{m}(C_m^i \cdot i \cdot (i-1)) \cdot T^4) = O(|Y| \cdot (m^2 - m) \cdot 2^{m-2} \cdot T^4)$ time to compute the utility at all the station nodes, without re-using any information. We find that the probability of transit line j arriving after α time given that the passenger has already waited r time at station y, which appears in Definition 4 reads: $1 - \sum_{\alpha=0}^{\theta} w_y^j(\alpha, r) = \frac{1-\sum_{\alpha=0}^{r+\theta} w_y^j(\alpha,0)}{1-\sum_{\alpha=0}^{r} w_y^j(\alpha,0)}$. In our implementation, we use $O(|Y| \cdot m \cdot T)$ memory to store $1 - \sum_{\alpha=0}^{\theta} w_y^j(\alpha, 0), \forall \theta \leq T, y \in Y, j \in M_y$. Then, the time complexity for computing utility for all station nodes becomes $O(|Y| \cdot m^2 \cdot 2^m \cdot T^3)$. ■

In summary, the time complexity of the algorithm is $O(|Y| \cdot m^2 \cdot 2^m \cdot T^3)$. Considering that there is a constant maximum number of transit lines at a station, the time complexity is $O(|Y| \cdot T^3)$, which leads to a pseudo-polynomial time algorithm in $|Y|$ and T.

5.4 Search space reduction

Although the DP approach is a pseudo-polynomial time algorithm in $|Y|$ and T, the computation time can still be high in large-scale networks when T is large. Therefore, we propose some search space reduction techniques to further de-

crease the computation time in practice.

5.4.1 Eliminate the infeasible paths

The simplest pruning technique we employ is to eliminate infeasible paths, i.e., paths that have zero probability of being used based on the minimum realizable travel time on each link. This pruning can be performed by simply running a shortest path search on a modified graph where the link weight is the minimum realizable travel time, to find the shortest path distance from each station to the destination. Assume that the minimum realizable travel time from station y to the destination is α_y. The utility for any node at station y given a time budget smaller than α_y will be zero.

This method is similar to the pruning method in [96] for road networks. Since the complexity of the shortest path algorithm is dominated by the complexity of the SOTA problem, the cost of the pruning method is negligible compared to the total computation time.

5.4.2 Search space reduction using transit line dominance

To compute the utility at an arrival node, we need to compare the utility of boarding the transit service (line node) and continuing to wait (station node). In this section, we propose a definition of transit line dominance, and a set of computationally efficient conditions to check for such dominance. This allows us to save the computation for the utility of the corresponding station node in the case of a dominating line node.

Dominance definition and properties

Definition 5. *Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t, r). Transit line i dominates X if $u_{L_y^i}(t) \geq u_{S_y^{X\setminus i}}(t, r)$. We use $i \succeq X$ to represent that i dominates X, and $i \prec X$ to represent that i does not dominate X.*

According to the definition, if $i \succeq X$, the passenger should board the transit line i; If $i \prec X$, the passenger should continue to wait for the transit lines in X. We now propose a useful claim that will be used in the proof of the dominance properties that we propose later.

Claim 7. *Assume that the passenger is at station y with passenger state (t, r). $u_{S_y^X}(t, r) \leq u_{S_y^{X'}}(t, r), \forall X' \supset X$.*

Proof. The passengers in our system are utility maximizers. Therefore, more transit line choices will make the utility of the station node increase or remain the same. ■

Assume that the passenger is at station y with passenger state (t, r). If $i \succeq X$, then $i \succeq X', \forall X' \subset X$. Correspondingly, if $i \prec X$, then $i \prec X', \forall X' \supset X$.

Proof. We prove the first part of the claim. The second part can be proved using similar reasoning. Assume that the passenger is at station y. According to Definition 5, $u_{L_y^i}(t) \geq u_{S_y^X}(t, r)$ if $i \succeq X$. In addition, for any $X' \subset X$, $u_{S_y^{X'}}(t, r) \leq u_{S_y^X}(t, r)$ based on Claim 7. Therefore, $u_{L_y^i}(t) \geq u_{S_y^{X'}}(t, r)$, and thus $i \succeq X \implies i \succeq X', \forall X' \subset X$. ■

Assume that the passenger is at station y with passenger state (t, r). If $u_{L_y^i}(t) \geq$

$u_{L_y^j}(t)$, then: (1) $i \succeq X$ for any X such that $j \succeq X$; (2) $j \prec X'$ for any X' such that $i \prec X'$.

Proof. For (1), if $j \succeq X$, then $u_{L_y^j}(t) \geq u_{S_y^X}(t, r)$ according to the definition of transit line dominance. Since $u_{L_y^i}(t) \geq u_{L_y^j}(t)$, we derive $u_{L_y^i}(t) \geq u_{S_y^X}(t, r)$, which proves that $i \succeq X$.

For (2), as $i \prec X'$, $u_{L_y^i}(t) \leq u_{S_y^X}(t, r)$. Since $u_{L_y^i}(t) \geq u_{L_y^j}(t)$, $u_{L_y^j}(t) \leq u_{S_y^X}(t, r)$, which represents $j \prec X'$. ■

According to the two properties above, we can infer transit line dominance based on the transit line dominance that we already know. Therefore, at an arrival node $A_y^{i,X}$ with passenger state (t, r), if we can infer $i \succeq X$ from the known dominance, we can reduce the computation for $u_{S_y^X}(t, r)$.

Dominance conditions

In this section, we present a series of conditions that can be assessed efficiently, and are practically useful to reduce the number of utility computations at station nodes. The conditions are considered at arrival nodes in order.

The first dominance condition is a subproblem pruning technique. If the condition is satisfied, we do not need to compute the utility at the corresponding station node at this specific passenger state.

Proposition 1. *(Subproblem pruning) Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t, r). If $u_{L_y^i}(t) \geq \max_{j \in X}\{u_{L_y^j}(t-1)\}$, then $i \succeq X$.*

Proof. The utility function is a nondecreasing function with respect to the time budget, i.e., $u_{L_y^j}(t-1) \geq u_{L_y^j}(t'), \forall t' \leq t-1, \forall j \in X$. In addition, $u_{S_y^X}(t,r)$ is a weighted average of $u_{L_y^j}(t')$ for $t' \leq t$ and $j \in X$ where the sum of the weight is not larger than 1. Therefore, if the condition is satisfied, $u_{L_y^i}(t) \geq u_{S_y^X}(t,r) \implies i \geq X$. ■

Proposition 1 needs $O(m)$ time to be checked. The intuition is as follows: Assume that we know all the transit lines in X will arrive right after i arrives. If boarding the transit line i under this assumption has a larger utility than waiting for the rest, then boarding the transit line i also has a larger utility than waiting for the rest without the assumption (i.e. the transit lines in X might arrive later).

If Proposition 1 cannot prove that $i \geq X$, then we need to compute $u_{S_y^X}(t,r)$ explicitly. We show next that, in some cases, we can safely remove some transit lines in X and only consider a subset of the candidate transit lines. Before we propose this candidate set pruning technique, we first provide the following definition of *improving lines* that will be used later on.

Definition 6. *Assume that the passenger is at station y with passenger state (t,r). Transit line k improves line j at station y if $u_{S_y^{\{k,j\}}}(t,r) > u_{S_y^{\{j\}}}(t,r)$.*

A sufficient and necessary condition for transit line k improving line j at station y is given as follows.

Claim 8. *Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t,r). Line k improves line j if there exists some state $(t-\gamma, r+\gamma)$ such that $u_{L_y^k}(t-\gamma) > \sum_{\theta=1}^{t-\gamma}(w_y^j(\theta, r+\gamma) \cdot u_{L_y^j}(t-\gamma-\theta))$.*

Proof. If there exists a passenger state $(t-\gamma, r+\gamma)$ satisfying the condition above in Claim 8, the passenger should board k when it arrives at state $(t-\gamma, r+\gamma)$

instead of continuing to wait for j. Therefore, the utility at the station node with $\{k, j\}$ is larger than the utility at station node with only line j. ■

Now, we propose the candidate set pruning technique as follows.

Proposition 2. *(Candidate set pruning) Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t, r). Let $Z = \{j \in X \mid u_{L_y^i(t)} < u_{L_y^j}(t-1)\}$. When we compute $u_{S_y^X}(t, r)$, we can instead compute $u_{S_y^{X'}}(t, r)$ where $X' \subseteq X$ and $X' = Z \cup \{j \in X \mid$ j improves at least one line in $Z\}$.*

Proof. Let $K = X \backslash X'$. Since no line in K improves the lines in Z, it is never better to take a line in K than waiting for the lines in Z, i.e., $u_{S_y^{Z\cup K}}(t, r) = u_{S_y^Z}(t, r)$. Since $Z \subset X'$, it is also never better to take the line in K than waiting for the lines in X', which implies $u_{S_y^{X'}}(t, r) = u_{S_y^X}(t, r)$. ■

We can consider the candidate set pruning technique as an extension of Proposition 1 since Z is obtained when we check the condition in Proposition 1. If $|Z| = 0$, the condition in Proposition 1 is satisfied, $i \succeq X$. Otherwise, we prune the transit lines in $\{j \in X \backslash Z \mid j$ cannot improve any line in $Z\}$ before computing for the utility at the station node.

According to Definition 4, we iterate through the time intervals $\{z \mid 1 \leq z \leq t\}$ when we compute $u_{S_y^X}(t, r)$. Here z is the waiting time for the first transit line arrival. In some cases, we can infer $i \succeq X$ when we finish the computation for an iteration $z < t$ and thus reduce the computation for the rest of the iterations.

Proposition 3. *(Time interval pruning) Assume that the passenger is at $A_y^{i,X}$ with passenger state (t, r). Let u_{max} be the maximum utility among all subsequent arrival nodes of S_y^X at passenger state $(t - z, r + z)$, i.e., $u_{max}(z) = \max_{j \in X}\{u_{A_y^{j,X\backslash j}}(t - z, z + r)\}$. Let $p_{j,X}(\theta)$*

be the probability that transit service j is the first transit service arrives at the station among all transit lines in X, and it arrives at the station at passenger state $(t-\theta, r+\theta)$, i.e., $p_{j,X}(\theta) = w_y^j(\theta, r) \prod_{k \in X \setminus j} (1 - \sum_{\alpha=1}^{\theta} w_y^k(\alpha, r))$. Let $u_{sum}(z)$ be the expected utility for the cases where there is one transit service in X arriving at the station from passenger state (t, r) to $(t-z, r+z)$, i.e., $u_{sum}(z) = \sum_{j \in X} \sum_{\theta=1}^{z} (p_{j,X}(\theta) \cdot u_{A_y^{j,X\setminus j}}(t-\theta, \theta+r))$. If there exists $z \leq t$ such that $u_{L_y^i}(t) \geq u_{sum}(z) + (1 - \sum_{j \in X} \sum_{\theta=1}^{z} p_{j,X}(\theta)) \cdot u_{max}(z)$, then $i \geq X$.

Proof. If we can show that $u_{sum}(z) + (1 - \sum_{j \in X} \sum_{\theta=1}^{z} p_{j,X}(\theta)) \cdot u_{max}(z) \geq u_{S_y^X}(t, r)$, then the proposition is proved. According to Definition 4, $u_{S_y^X}(t, r) = u_{sum}(t)$. We can infer that $u_{S_y^X}(t, r) = u_{sum}(z) + \sum_{j \in X} \sum_{\theta=z+1}^{t} (p_{j,X}(\theta) \cdot u_{A_y^{j,X\setminus j}}(t-\theta, \theta+r))$. Since the utility function is non decreasing, we know that $u_{\max}(z) = \max_{j \in X, z \leq \theta \leq t} \{u_{A_y^{j,X\setminus j}}(t-\theta, \theta+r)\}$. In addition, $1 - \sum_{j \in X} \sum_{\theta=1}^{z} p_{j,X}(\theta) \geq \sum_{j \in X} \sum_{\theta=z+1}^{t} p_{j,X}(\theta)$. Therefore, we can conclude that the proposition is correct since $u_{S_y^X}(t, r) \leq u_{\text{sum}}(z) + (1 - \sum_{j \in X} \sum_{\theta=1}^{z} p_{j,X}(\theta) \cdot u_{\max}(z)$. ■

Whenever we finish computing for a specific time interval $z = i$, we can check the condition in Proposition 3. If the condition is not satisfied, we let $z = i + 1$ and continue the procedure. However, if the condition is satisfied, we know that $i \geq X$ and thus can reduce the computation for $i + 1 \leq z \leq t$.

To sum up all the techniques discussed in this section and Section 5.4.2, we modify the procedure of computing the utility at arrival nodes as follows. Two dictionaries *dom* and *nondom* are used to record the transit line dominance and non-dominance. *update_dom* and *update_nondom* are two functions to update *dom* and *nondom* according to dominance properties in Section 5.4.2. Specifically, for any arrival node $A_y^{i,X}$, the pruning is done via the following set of steps.

Step 1: Check if X is in $dom[i,t,r]$: if yes, return $u_{L_y^i}(t)$; Otherwise, go to Step 2.

Step 2: Check if X is in $nondom[i,t,r]$: if yes, compute $u_{S_y^X}(t,r)$ and return it; Otherwise, go to Step 3.

Step 3: Check if $i \succeq X$ or $i \prec X$ using the pruning techniques. If $i \succeq X$, run *update_dom* and return $u_{L_y^i}(t)$; Otherwise, run function *update_nondom* and return $u_{S_y^X}(t,r)$.

We call the algorithm DP with dominance. Note that all the methods discussed retain the optimality of the solution.

5.4.3 Heuristic rules

In this section, we propose three heuristic rules, which can be employed in the procedure of checking dominance, to further decrease the computation time. We will show in the numerical experiments that the heuristics can significantly reduce the computation time without much loss of the results accuracy.

Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t,r). Let $Z = \{j \in X \mid u_{L_y^i(t)} < u_{L_y^j}(t-1)\}$, and $p = \prod_{j \in Z} \sum_{\theta=1}^{t} I_{u_{L_y^i(t)} \geq u_{L_y^j(t-\theta)}} \cdot w_y^j(\theta, r)$ where $I_{u_{L_y^i(t)} \geq u_{L_y^j(t-\theta)}}$ is an indicator function. If $p \geq \epsilon$, we say that the passenger should board the transit line i.

This is an extension of Proposition 1. ϵ is a constant coefficient. p is the probability of the case that boarding transit line i has a larger utility than boarding any other single transit line in X. If p is large, then only with a small probability, the realization of the waiting time for a transit line in X can make its utility larger than boarding transit line i.

Assume that the passenger is at node $A_y^{i,X}$ with passenger state (t, r). If $u_{L_y^i}(t) \geq \sum_{\theta=1}^{t} w_y^j(\theta, r) \cdot u_{L_y^j}(t-\theta), \forall j \in X$, we say that the passenger should board the transit line i.

In other words, the passenger should board transit line i if boarding it has a larger utility than waiting for any single line in X. It is an approximation since $u_{S_y^X}(t, r) \geq u_{S_y^{\{j\}}}(t, r), \forall j \in X$ according to Claim 7. Therefore, it is possible that $u_{S_y^X}(t, r) > u_{L_y^i}(t)$.

Assume that the passenger is at $A_y^{i,X}$ with passenger state (t, r). Let $u_{max}(z) = \max_{j \in X}\{u_{A_y^{j,X\setminus j}}(t-z, z+r)\}$, and $u_{sum}(z) = \sum_{j \in X}(\sum_{\theta=1}^{z} w_y^j(\theta, r) \prod_{k \in X\setminus j}(1 - \sum_{\alpha=1}^{\theta} w_y^k(\alpha, r)) \cdot u_{A_y^{j,X\setminus j}}(t-\theta, \theta + r))$. If there exists $z \leq t$ such that $\beta \cdot u_{L_y^i}(t) \geqslant u_{sum}(z) + (1 - \sum_{j \in X}(\sum_{\theta=1}^{z} w_y^j(\theta, r) \prod_{k \in X\setminus j}(1 - \sum_{\alpha=1}^{\theta} w_y^k(\alpha, r)))) \cdot u_{max}(z)$, then we say that the passenger should board transit line i.

This is an extension of Proposition 3. The difference is that we add a constant relaxation coefficient β ($\beta > 1$) in the condition. This represents that when we compute the utility of a station node, if β times the utility of the line node is larger than the largest possible utility of the station node, we say that the passenger should board the transit line that arrives. β in Heuristic 5.4.3 and ϵ in Heuristic 5.4.3 are used to control the tradeoff between the results accuracy and computation performance. In the numerical experiments, we let $\beta = 1.25$ and $\epsilon = 0.75$.

In the following section, we present numerical results and compare the three versions of the solution algorithm introduced in this work: (1) DP; (2) DP with dominance; (3) DP with dominance and heuristics.

5.5 Numerical experiments

In this section, we present numerical experiments focused on illustrating the runtime performance of the various versions of the solution, as well as the practical value of using a SOTA policy in transit networks. All the algorithms are coded in Python 3.6, and all tests are conducted on an AMD Ryzen processor computer (3.4 gigahertz, 16 gigabytes RAM). We first validate the algorithms and conduct a sensitivity analysis in a controlled setting using a synthetic transit network, and then test them on the Chicago transit network.

5.5.1 Estimating travel time and headway distributions

To obtain the required network and transit line information, we develop a procedure for taking the input data (GTFS data) and generate all the processed data we need in the experiments, including travel time distributions, headway distribution, and waiting time distributions. The first step is to generate the travel time distributions for each transit line. Studies have shown that travel time distributions on road networks can be well approximated by a lognormal distribution (see [28, 52, 123]). Therefore, in our experiments, we model the travel time distributions using appropriately parameterized lognormal distributions. A lognormal distributed random variable X is parameterized by two parameters μ and σ that are, respectively, the mean and standard deviation of the variable's natural logarithm. For every two adjacent stations S_1 and S_2 in a transit line, we calibrate μ and σ for the travel time between the two stations based on the following steps:

(1) Compute the minimum realizable travel time t_m by dividing the distance between S_1 and S_2 by the corresponding speed limit.

(2) Let the mode of the lognormal distributed variable, i.e., the point of the maximum of the probability density function, be the difference between the scheduled departure time at S_2 and S_1. The mode of a lognormal distribued variable is $e^{\mu-\sigma^2}$. Therefore, if the schedule departure time at S_1 and S_2 are 8:45 and 9:00 and t_m = 5 min, then we have $e^{\mu-\sigma^2}$ = 10.

(3) In the absence of more information from GTFS, σ of the lognormal distribution is sampled uniformly from $0.25 \leq \sigma \leq 0.5$ so that the variance is neither too large or too small.

(4) Compute μ according to the mode and σ. Shift the distribution rightwards by t_m.

As σ is randomly sampled from a given range, a sensitivity analysis is conducted in Section 5.5.2. If one can get access to the transit vehicle's real-time location, σ can be obtained from the real realized travel time data. After computing the travel time distribution between every two adjacent stations, we can obtain the travel time distribution between the origin and all other stations on each transit line by computing the appropriate summation of the random variables due to the independence assumption.

The headway at the origin station of each transit line is set to the mean of the difference between the scheduled departure time for every two consecutive trips in our experiment time span. For instance, if the scheduled departure times for a transit line *i* at the origin station are 8:45, 8:55, 9:10, then we let the headway be $\frac{10+15}{2}$ = 12.5. The headway for other stations is computed as follows. Let

X_1 and X_2 be the arrival time of the first transit vehicle and the second transit vehicle of the transit line of interest at a station S. O is the origin, and h is the deterministic headway at O. The cumulative distribution function of the headway is computed as follows:

$$P(X_2 - X_1 \leq t) = P(X_2 \leq X_1 + t) = \int_0^\infty P(X_2 \leq x + t) \cdot P(X_1 = x) dx$$

where $P(x_2 \leq x + t)$ and $P(X_1 = x)$ are the cumulative distribution function of X_2 and the probability density function of X_1. Let $t_{O,S}$ be the travel time between O and S. The distribution of X_1 is the same as the distribution for $t_{O,S}$, and the distribution of X_2 is to shift the distribution for $t_{O,S}$ rightwards by h.

Finally, we estimate the waiting time distribution from the headway distributions following [57]. Let $\mathbb{E}_{i,y}[h_{i,y}]$ be the expected headway of transit line i at station y, and $H_{i,y}(\cdot)$ be the cumulative distribution function of headway of transit line i at station y. Then the waiting time reads:

$$w_y^i(t, 0) = \frac{1}{\mathbb{E}_{i,y}[h_{i,y}]} \cdot (1 - H_{i,y}(t))$$

5.5.2 Synthetic network experiments

The synthetic network we use is a 3-line 3-station transit network, as shown in Figure 5.3. The three transit lines pass through each of the three stations. The line attributes are presented in Table 5.2. The headway and travel time are in minutes.

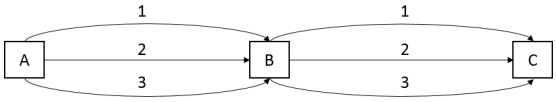

Figure 5.3: A 3-line synthetic transit network.

Line i	Headway at station A	Travel time from A to B	Travel time from B to C
1	10	4	5
2	15	4	3
3	12	7	4

Table 5.2: Line attributes of the 3-line synthetic transit network. All times are provided in minutes.

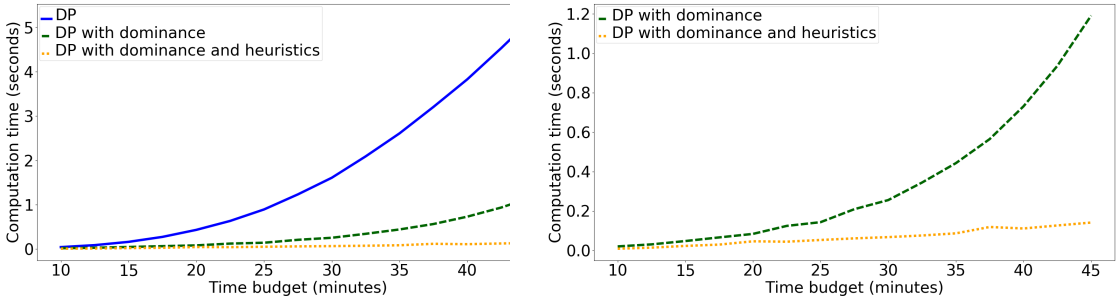

Figure 5.4: Computation time of three algorithms on the synthetic network. *Left:* All three algorithms. *Right:* Only algorithms with the dominance based pruning to showcase the relative improvement.

Performance analysis

We first illustrate the performance of the algorithms, specifically the relationship between the computation time and the time budget. Let station A be the origin, and station C be the destination, and the time discretization of the algorithm to be 15 seconds. The three algorithms introduced in Section 5.3 are tested on time budgets ranging from 10 min to 45 min with a step size of 2.5 min.

As shown in Figure 5.4, the algorithms with search space reduction techniques perform significantly better than the basic DP approach. The DP with the dominance test can provide an average computation time reduction of 77.8% compared to the basic DP approach. The heuristics further decrease the computation time by 68.9% on average relative to the DP with dominance. The time reduction percentage ($\frac{\text{Computation time of DP - Computation of DP with dominance}}{\text{Computation time of DP}}$) is typi-

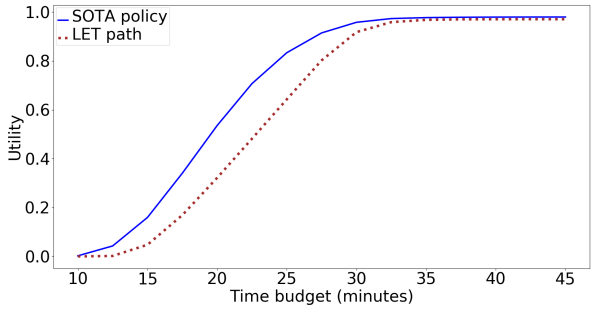

Figure 5.5: Comparison of the utility between the SOTA policy and LET path.

cally higher when the time budget is large. When the time budget is 45 min, the time reduction can be 97.3% for the DP with dominance and heuristics. The average relative error for the heuristic method is only 2.8% in this experiment. Therefore, the heuristic (at least in this case) provides a satisfactory trade-off between accuracy and computation time.

We also quantify the benefits of utilizing a SOTA policy instead of using the least expected travel time (LET) path. This is done by using the expected value of both the travel time and waiting time distributions to compute the LET path between station A and station C. The utility of the LET solution is then computed based on the probability of success when using the LET path for different time budgets. Figure 5.5 illustrates the utility difference between the LET path and the SOTA policy we obtain from our algorithm. When the time budget is very small, the utility difference is zero because the passenger cannot reach the destination on time regardless of the policy the passenger uses. Similarly, when the time budget is very large, the utility difference is also 0 because the passenger reaches the destination on time with high probability even with a sub-optimal route. The SOTA policy will always be no worse than the LET solution by definition. The maximum utility difference observed in this experiment is 23 percent (when the time budget is 22.5 min in this case).

Sensitivity to the modes of the travel time distributions.

In this section, we analyze the effect of the modes[3] of travel time distributions on computation time. The modes of the travel time distributions on each link are sampled uniformly from a range instead of being predefined. Two sets of experiments are conducted. In the first set of experiments, we let the width of the range be 1. More specifically, we first generate a random number $i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ uniformly, and then set the range to be $[i, i+1)$. Assume that X_1 and X_2 are two random variables sampled uniformly from this range. Then, $E(|X_1 - X_2|) = \frac{1}{3}$ since $|X_1 - X_2|$ follows a triangle distribution with the parameters $a = 0$, $c = 0$ and $b = 1$. In the second set of experiments, the range is set to $[1, 10)$, and $E(|X_1 - X_2|) = 3$ since $|X_1 - X_2|$ follows a triangle distribution where $a = 0$, $c = 0$ and $b = 9$. We call the first set of experiments *low diff*, and the second set of experiments *high diff*. Each set of experiments are conducted for 100 times.

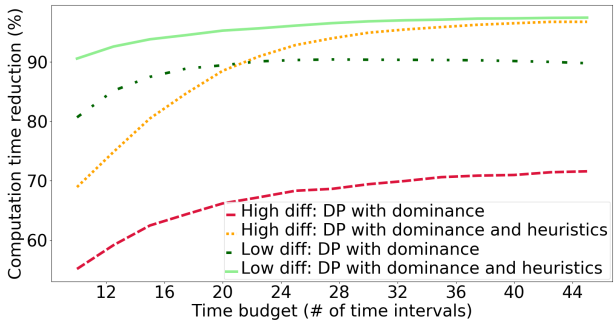

Figure 5.6: Computation time reduction of speed-up techniques on two cases. *high diff* represent the case where the travel time distributions' modes are highly different, while *low diff* represents the case where the travel time distributions' modes are relatively close.

The computation time reduction with respect to the time budget is shown in Figure 5.6. The search space reduction techniques work well in both sets of experiments. The average computation time reductions for the DP with dom-

[3]Note that *mode* here is the statistical definition.

inance are 88.9% and 67.1% for the *low diff* and *high diff* cases respectively relative to standard DP, and the time reductions are 95.7% and 89.8% for the DP with dominance and heuristics. The search space reduction techniques work better in the *low diff* cases because the transit services that arrive the first will intuitively dominate the other transit lines more often with similar travel time distributions, since waiting is not likely to increase the utility. Another interesting result is that the heuristics will provide a higher time reduction for the *high diff* cases compared to the DP with dominance algorithm. A potential reason is that a line will have a higher probability of being better than any other single candidate line at the same station in this case, which indicates dominance according to the Heuristic 5.4.3.

Sensitivity to the σ parameter

We now analyze the sensitivity of the algorithm performance to the parameter σ in the lognormal travel time distributions. Assume that the CDF of a lognormal distribution is $F(x)$, and γ is the mode of the distribution. In Section 5.5.2, we use $\sigma = 0.25$, which makes $F(1.5 \cdot \gamma) \approx 0.9$ when γ is in the range shown in Table 5.2. To show the effects of σ on the distribution, we compare the distribution with $\sigma = 0.25$ and $\sigma = 0.5$ in Figure 5.7. When $\sigma = 0.5$, $F(1.5 \cdot \gamma) \approx 0.6$.

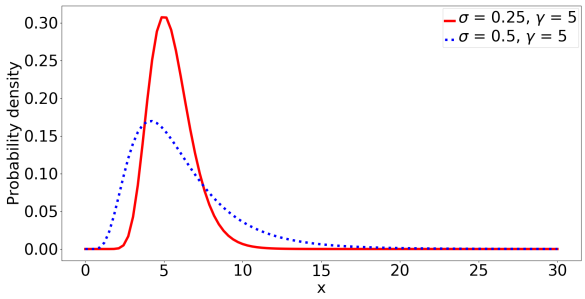

Figure 5.7: The PDF of lognormal distributions with different σ.

To analyze the sensitivity of σ on the algorithm's performance, two sets of experiments are conducted. In the first set of experiments, the algorithms are tested under $\sigma = 0.5$. In the second set of experiments, σ is uniformly sampled from $[0.25, 0.5]$ for each link of each transit line. Again, each set of experiments are conducted for 100 times.

Table 5.3: Algorithms' performance under different σ settings.

Parameter setting	Computation time reduction for DP with dominance	Computation time reduction for DP with dominance and heuristics
$\sigma = 0.25$	77.8%	92.6%
$\sigma = 0.5$	80.3%	94.1%
σ randomly chosen from $[0.25, 0.5]$	79.8%	93.9%

As shown in Table 5.3, the search space reduction techniques perform similarly under different σ values. The computation time is reduced slightly more when the variance of the travel time distribution is larger. In the following experiments for the Chicago transit network, we use σ randomly sampled from $[0.25, 0.5]$.

5.5.3 Chicago network experiments

We now use the Central/South region (including downtown) of the Chicago transit network as a case study. The network is created according to the GTFS data published by the Chicago Transit Authority (CTA) through Google's GTFS project. The GTFS data contains schedules and associated geographic information. In the experiments, only active bus lines in the morning (6 am to 10 am) are considered. The transit network, which contains 49 transit lines and 1565 stations, is shown in Figure 5.8.

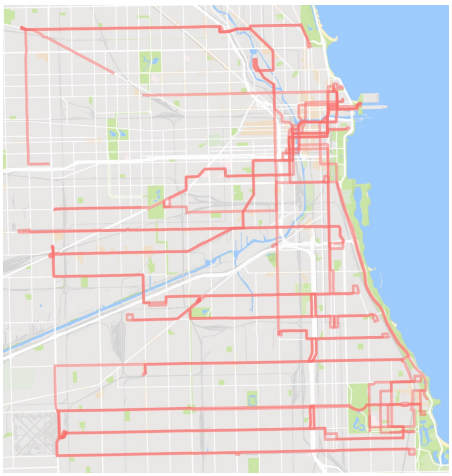

Figure 5.8: Chicago transit network (Central/South region).

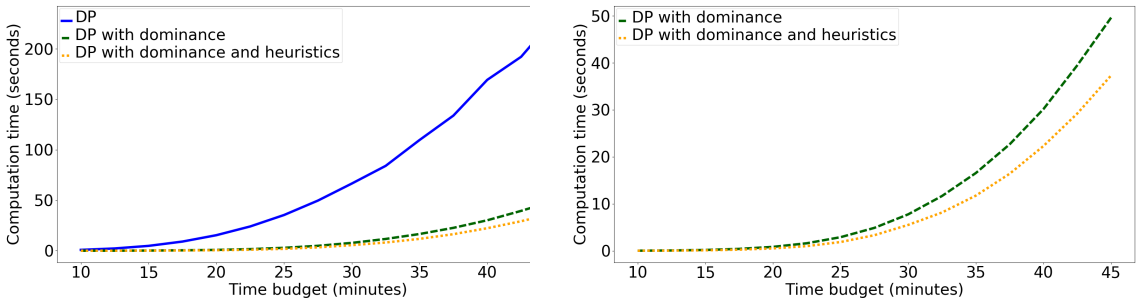

Figure 5.9: Computation time of three algorithms on the Chicago transit network. *Left:* All three algorithms. *Right:* Only algorithms with the dominance based pruning to showcase the relative improvement.

Runtime performance in the Chicago network

The average travel time to commute in the United States is 26.1 min according to the U.S. Census Bureau [16]. Therefore, we select 100 ODs randomly from the station set under the constraints: (1) The expected trip duration is from 15 min to 45 min; (2) At least one of the origin and the destination is in the downtown area, to simulate the case of commuting. The experiments are conducted for the time budgets which also span from 10 min to 45 min. For each time budget, we use the same three algorithms described previously to compute the utility. The computation time for each algorithm is shown in Figure 5.9.

The DP with dominance reduces the computation time by 89.3% on average. The computation time reductions are significant, but not as high as in the synthetic network shown in Figure 5.3 especially when T is large. In the synthetic network, there are only three stations which limit the search space. However, in real-size networks, the search space will be much larger since there are more stations and transit lines to be considered. The heuristic rules further decrease the computation time by 34.0% on average relative to the DP with dominance. The average relative error for the heuristic rules is about 1.6%.

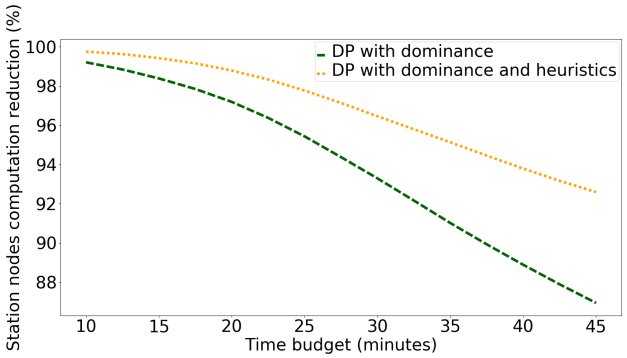

Figure 5.10: The computation reduction for computing the utility on station nodes. The vertical axis shows the percentage of function calls reduction to compute the utility at stations.

Recall from Section 5.4, if $i \succeq X$ at an arrival node $A_y^{i,X}$ given passenger state (t, r), we save the computation for $u_{S_y^X}(t, r)$. In the experiments, we also record the number of function calls to compute the utility at station nodes. Figure 5.10 shows the percentage of the function calls reduction by DP with dominance and DP with dominance and heuristics. Both algorithms provide more than 90% function calls reduction for all the cases. However, the reduction decreases as the time budget increases. When the time budget is higher, the set of feasible arrival nodes that followed by a station node S_y^X at passenger state (t, r) is larger. Here an arrival node $A_y^{i,X}$ is *feasible* at passenger state (t, r) if it is not pruned by the techniques in Section 5.4. Let Z_y^X be the set of feasible arrival nodes followed

by station node S_y^X. To save the computation for $u_{S_y^X}(t, r)$, every line in $\{i \mid A_y^{i,X} \in Z_y^X\}$ needs to dominate X, which is of less probability when $|Z|$ is higher.

The average computation time of the DP with dominance and heuristics is only about 9.2 seconds on a personal computer, which indicates the potential of running the algorithm in real-time applications.

Compare the utility of the SOTA policy and LET path

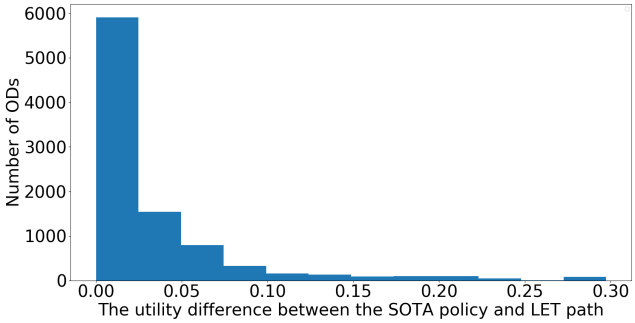

Figure 5.11: The histogram of the utility difference between the SOTA policy and LET path on the Chicago transit network.

Recall from Section 5.5.2, the utility difference between the SOTA policy and LET path can be as large as 0.23 in the synthetic network. In this section, we test the utility difference in the Chicago transit network. Intuitively, if there is one and only one transit line that can directly take the passenger to the destination without transferring, the SOTA policy and LET path will likely be the same due to the time cost of transferring. Therefore, we compare the utility difference of all the ODs from the station set under the constraints: (1) The expected trip duration is from 15 min to 45 min; (2) At least one of the origin and the destination is in the downtown area; (3) There is no transit line or there are more than one transit line that can directly take the passenger to the destination. The number

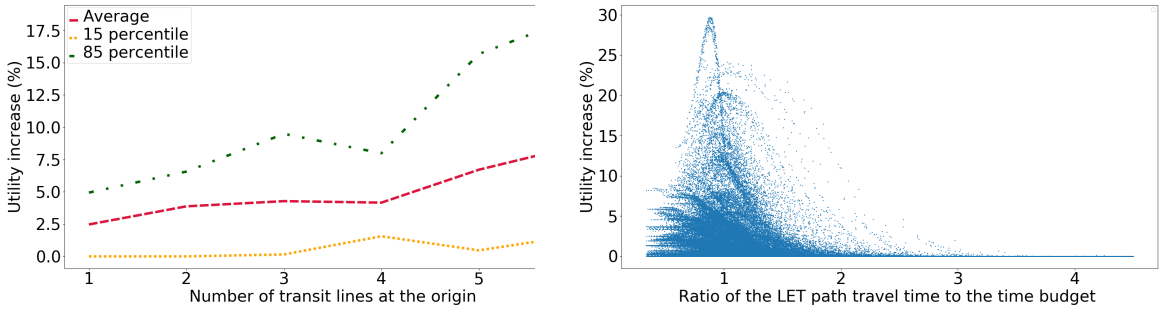

Figure 5.12: The utility improvement of the SOTA policy over the LET path under different circumstances. *Left:* With respect to the number of transit lines passing by the origin. *Right:* With respect to the ratio of the travel time of the LET path and the time budget.

of ODs satisfying the above conditions is 9290. Again, the experiments are conducted for the time budgets from 10 min to 45 min. For each OD, we record the maximum utility difference between the SOTA policy and LET path among all the time budgets we test. Figure 5.11 shows the histogram of the maximum utility difference. The utility difference for 19.6% of the ODs is larger than 0.05, and the utility difference for 7.6% of the ODs is larger than 0.1. The utility difference is close to 0 for most of the ODs, which are the cases that either there is only one route choice between the origin and the destination or there is one obvious better line than the other choices. It should be noted that the results highly depend on the network and the travel time distribution we use.

To give insights of what trips will likely benefit from the SOTA policy, we show the utility increases of the SOTA policy over the LET path on different circumstances with respect to: i) The number of transit lines passing by the origin; ii) The ratio of the least expected time (LET) path travel time to the time budget. As shown in Figure 5.12 (Left), a SOTA policy outperforms the LET path more when the number of transit lines at the origin increases, which demonstrates that a SOTA policy is more advantageous when there are more states and op-

tions to consider. In addition, even though there might be more transit lines to transfer on the way, it may not be beneficial to conduct a transfer in most of the cases due to the transfer time. Therefore, when there is only one transit line passing by the origin, the average utility difference is about 0.025, which is close to 0. The ratio of the LET path travel time to the time budget represents how sufficient the time budget is relative to the trip length. Figure 5.12 (Right) shows the utility increase versus the ratio. If the ratio is relatively large (larger than 2), passengers cannot increase the utility by using the adaptive routing policy since it is difficult to get to the destination in a very limited time budget; On the other hand, if the ratio is very small (smaller than 0.5), the utility increase is also close to 0 since passengers will likely reach the destination within the given time budget whatever transit line they board. The SOTA policy performs the best when the ratio is around 1. In summary, the SOTA policy in transit networks shines the most when there are more states to consider and when the time budget is neither too high or too low relative to the trip length.

BIBLIOGRAPHY

[1] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[2] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment -supplemental material-. 2017.

[3] Javier Alonso-Mora, Alex Wallar, and Daniela Rus. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3583–3590. IEEE, 2017.

[4] Mohammad Asghari and Cyrus Shahabi. Adapt-pricing: a dynamic and predictive technique for pricing to maximize revenue in ridesharing platforms. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2018.

[5] Carlos Lima Azevedo, Katarzyna Marczuk, Sebastian Raveau, Harold Soh, Muhammad Adnan, Kakali Basak, Harish Loganathan, Neeraj Deshmunkh, Der-Horng Lee, Emilio Frazzoli, et al. Microsimulation of demand and supply of autonomous mobility on demand. *Transportation Research Record: Journal of the Transportation Research Board*, (2564):21–30, 2016.

[6] Rounaq Basu, Andrea Araldo, Arun Prakash Akkinepally, Kakali Basak, Ravi Seshadri, Bat Hen Nahmias Biran, Neeraj Deshmukh, Nishant Kumar, Carlos Lima Azevedo, and Moshe Ben-Akiva. Implementation & policy applications of amod in multi-modal activity-driven agent-based urban-simulator simmobility. 2018.

[7] Reinhard Bauer and Daniel Delling. Sharc: Fast and robust unidirectional routing. *Journal of Experimental Algorithmics (JEA)*, 14:4, 2009.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[9] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions

for vision architectures. In *International Conference on Machine Learning*, pages 115–123, 2013.

[10] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[11] Patrick M Boesch, Francesco Ciari, and Kay W Axhausen. Autonomous vehicle fleet sizes required to serve different levels of demand. *Transportation Research Record: Journal of the Transportation Research Board*, (2542):111–119, 2016.

[12] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[13] Chris Brownell and Alain Kornhauser. A driverless alternative: fleet size and cost requirements for a statewide autonomous taxi network in new jersey. *Transportation Research Record: Journal of the Transportation Research Board*, (2416):73–81, 2014.

[14] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

[15] Juan Camilo Castillo, Dan Knoepfle, and Glen Weyl. Surge pricing solves the wild goose chase. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 241–242. ACM, 2017.

[16] census.gov. Average one-way commuting time by metropolitan areas. `https://www.census.gov/library/visualizations/interactive/travel-time.html`, 2017.

[17] Bi Yu Chen, William HK Lam, Agachai Sumalee, Qingquan Li, and Mei Lam Tam. Reliable shortest path problems in stochastic time-dependent networks. *Journal of Intelligent Transportation Systems*, 18(2):177–189, 2014.

[18] Mengjing Chen, Weiran Shen, Pingzhong Tang, and Song Zuo. Optimal vehicle dispatching schemes via dynamic pricing. *arXiv preprint arXiv:1707.01625*, 2017.

[19] Peng Will Chen and Yu Marco Nie. Optimal transit routing with partial online information. *Transportation Research Part B: Methodological*, 72:40–58, 2015.

[20] ChoiceMetrics. Ngene 1.0 user manual and reference guide: The cutting edge in experimental design, 2014.

[21] Roberto Cominetti and José Correa. Common-lines and passenger assignment in congested transit networks. *Transportation Science*, 35(3):250–267, 2001.

[22] Hussein Dia and Farid Javanshour. Autonomous shared mobility-on-demand: Melbourne pilot simulation study. *Transportation Research Procedia*, 22:285–296, 2017.

[23] Shadi Djavadian and Joseph YJ Chow. An agent-based day-to-day adjustment process for modeling 'mobility as a service'with a two-sided flexible transport market. *Transportation research part B: methodological*, 104:36–57, 2017.

[24] Shadi Djavadian and Joseph YJ Chow. Agent-based day-to-day adjustment process to evaluate dynamic flexible transport service policies. *Transportmetrica B: Transport Dynamics*, 5(3):281–306, 2017.

[25] B Donovan and DB Work. New york city taxi trip data (2010–2013), 2014.

[26] Brian Donovan and Daniel B Work. Using coarse gps data to quantify city-scale transportation system resilience to extreme events. *arXiv preprint arXiv:1507.06011*, 2015.

[27] Chinmoy Dutta. When hashing met matching: Efficient search for potential matches in ride sharing. *arXiv preprint arXiv:1809.02680*, 2018.

[28] Emam Emam and Haitham Ai-Deek. Using real-life dual-loop detector data to develop new methodology for estimating freeway travel time reliability. *Transportation Research Record: Journal of the Transportation Research Board*, (1959):140–150, 2006.

[29] Daniel J Fagnant and Kara M Kockelman. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, 40:1–13, 2014.

[30] Daniel J Fagnant and Kara M Kockelman. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. *Transportation*, 45(1):143–158, 2018.

[31] Daniel J Fagnant, Kara M Kockelman, and Prateek Bansal. Operations of shared autonomous vehicle fleet for austin, texas, market. *Transportation Research Record: Journal of the Transportation Research Board*, (2536):98–106, 2015.

[32] Yueyue Fan and Yu Nie. Optimal routing for maximizing the travel time reliability. *Networks and Spatial Economics*, 6(3-4):333–344, 2006.

[33] YY Fan, RE Kalaba, and JE Moore. Shortest paths in stochastic networks with correlated link costs. *Computers & Mathematics with Applications*, 49(9):1549–1564, 2005.

[34] YY Fan, RE Kalaba, and JE Moore II. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005.

[35] Cyrus Farivar. California's first proposed per-ride city tax to raise uber, lyft prices. `https://arstechnica.com/tech-policy/2018/03/oakland-wants-a-cut-of-uber-lyft-rides-mulls-states-first-per-ri` 2018.

[36] David Fiedler, Michal Čáp, and Michal Čertickỳ. Impact of mobility-on-demand on traffic congestion: Simulation-based study. *arXiv preprint arXiv:1708.02484*, 2017.

[37] Arthur Flajolet, Sébastien Blandin, and Patrick Jaillet. Robust adaptive routing under uncertainty. *Operations Research*, 66(1):210–229, 2017.

[38] H Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17(4):583–599, 1969.

[39] Song Gao and Ismail Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B: Methodological*, 40(2):93–122, 2006.

[40] Roman Garnett, Michael A Osborne, and Stephen J Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2010.

[41] Guido Gentile, Sang Nguyen, and Stefano Pallottino. Route choice on transit networks with online information at stops. *Transportation science*, 39(3):289–297, 2005.

[42] Raga Gopalakrishnan, Theja Tulabandhula, and Koyel Mukherjee. Sequential individual rationality in dynamic ridesharing. *Available at SSRN*, 2018.

[43] Ronald J Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. *ALENEX/ANALC*, 4:100–111, 2004.

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

[45] Ward Hanson and Kipp Martin. Optimizing multinomial logit profit functions. *Management Science*, 42(7):992–1003, 1996.

[46] Mingyang Hao and Toshiyuki Yamamoto. Analysis on supply and demand of shared autonomous vehicles considering household vehicle ownership and shared use. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 185–190. IEEE, 2017.

[47] Fang He, Xiaolei Wang, Xi Lin, and Xindi Tang. Pricing and penalty/compensation strategies of a taxi-hailing platform. *Transportation Research Part C: Emerging Technologies*, 86:263–279, 2018.

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

[49] Moritz Hilger, Ekkehard Köhler, Rolf H Möhring, and Heiko Schilling. Fast point-to-point shortest path computations with arc-flags. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:41–72, 2009.

[50] Sebastian Hörl, Alexander Erath, and Kay W Axhausen. Simulation of autonomous taxis in a multi-modal traffic scenario with dynamic demand. *Arbeitsberichte Verkehrs-und Raumplanung*, 1184, 2016.

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

[52] Timothy Hunter, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. Path and travel time inference from gps probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs*, 12(1), 2009.

[53] Frank Hutter and Michael A Osborne. A kernel for hierarchical parameter spaces. *arXiv preprint arXiv:1310.5738*, 2013.

[54] Renos Karamanis, Panagiotis Angeloudis, Aruna Sivakumar, and Marc Stettler. Dynamic pricing in one-sided autonomous ride-sourcing markets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3645–3650. IEEE, 2018.

[55] Nikita Korolko, Dawn Woodard, Chiwei Yan, and Helin Zhu. Dynamic pricing and matching in ride-hailing platforms. *Available at SSRN*, 2018.

[56] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[57] Richard C Larson and Amedeo R Odoni. *Urban operations research*. Number Monograph. 1981.

[58] Michael W Levin and Stephen D Boyles. A multiclass cell transmission model for shared human and autonomous vehicle roads. *Transportation Research Part C: Emerging Technologies*, 62:103–116, 2016.

[59] Dan Li and Evangelos Kanoulas. Bayesian optimization for optimizing retrieval systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 360–368. ACM, 2018.

[60] Hongmin Li and Woonghee Tim Huh. Pricing multiple products with the multinomial logit and nested logit models: Concavity and implications. *Manufacturing & Service Operations Management*, 13(4):549–563, 2011.

[61] Jie Li, Yu Sen Chen, Hao Li, Ingmar Andreasson, and Henk van Zuylen. Optimizing the fleet size of a personal rapid transit system: A case study in port of rotterdam. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 301–305. IEEE, 2010.

[62] Qianfei Li, Peng Will Chen, and Yu Marco Nie. Finding optimal hyperpaths in large transit networks with realistic headway distributions. *European Journal of Operational Research*, 240(1):98–108, 2015.

[63] Shuguang Liu, Weilai Huang, and Huiming Ma. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):434–445, 2009.

[64] Yang Liu, Prateek Bansal, Ricardo Daziano, and Samitha Samaranayake. A framework to integrate mode choice in the design of mobility-on-demand systems. *arXiv preprint arXiv:1805.06094*, 2018.

[65] Yang Liu, Prateek Bansal, Ricardo Daziano, and Samitha Samaranayake. A framework to integrate mode choice in the design of mobility-on-demand systems. `https://doi.org/10.1016/j.trc.2018.09.022`, 2018.

[66] Yang Liu and Samitha Samaranayake. Proactive rebalancing and speed-up techniques for on-demand high capacity vehicle pooling. *arXiv preprint arXiv:1902.03374*, 2019.

[67] Daniel J Lizotte, Tao Wang, Michael H Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[68] Ronald Prescott Loui. Optimal paths in graphs with stochastic or multi-dimensional weights. *Communications of the ACM*, 26(9):670–676, 1983.

[69] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE, 2013.

[70] Tai-Yu Ma. On-demand dynamic bi-/multi-modal ride-sharing using optimal passenger-vehicle assignments. In *Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), 2017 IEEE International Conference on*, pages 1–5. IEEE, 2017.

[71] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[72] Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2242–2249. IEEE, 2012.

[73] Katarzyna Anna Marczuk, Harold Soh Soon Hong, Carlos Miguel Lima Azevedo, Muhammad Adnan, Scott Drew Pendleton, Emilio Frazzoli, et al. Autonomous mobility on demand in simmobility: Case study of the central business district in singapore. In *Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2015 IEEE 7th International Conference on*, pages 167–172. IEEE, 2015.

[74] John Markoff. Google's next phase in driverless cars: No steering wheel or brake pedals. `https://www.nytimes.com/2014/05/28/technology/googles-next-phase-in-driverless-cars-no-brakes-or-steering-whee html`, 2014.

[75] Bertil Matérn. *Spatial variation*, volume 36. Springer Science & Business Media, 2013.

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

[77] Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4(4):347–365, 1994.

[78] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[79] Aditi Moorthy, Robert De Kleine, Gregory Keoleian, Jeremy Good, and Geoff Lewis. Shared autonomous vehicles as a sustainable solution to the last mile problem: A case study of ann arbor-detroit area. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 10(2017-01-1276), 2017.

[80] MTA. Metropolitan transportation authority ("mta") data feeds. `http://web.mta.info/developers/developer-data-terms.html#data`, 2018.

[81] William P Nanry and J Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, 2000.

[82] Yu Marco Nie and Xing Wu. Shortest path problem considering on-

time arrival probability. *Transportation Research Part B: Methodological*, 43(6):597–613, 2009.

[83] Mehrdad Niknami and Samitha Samaranayake. Tractable pathfinding for the stochastic on-time arrival problem. In *International Symposium on Experimental Algorithms*, pages 231–245. Springer, 2016.

[84] Evdokia Nikolova, Jonathan A Kelner, Matthew Brand, and Michael Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *European Symposium on Algorithms*, pages 552–563. Springer, 2006.

[85] Tim Nonner and Marco Laumanns. Shortest path with alternatives for uniform arrival times: Algorithms and experiments. In *OASIcs-OpenAccess Series in Informatics*, volume 42. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[86] Nurit Oliker and Shlomo Bekhor. A frequency based transit assignment model that considers online information. *Transportation Research Part C: Emerging Technologies*, 88:17–30, 2018.

[87] Axel Parmentier and Frédéric Meunier. Stochastic shortest paths and risk measures. *arXiv preprint arXiv:1408.0272*, 2014.

[88] Victor Picheny, Mickael Binois, and Abderrahmane Habbal. A bayesian optimization approach to find nash equilibria. *arXiv preprint arXiv:1611.02440*, 2016.

[89] George H Polychronopoulos and John N Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.

[90] Han Qiu, Ruimin Li, and Jinhua Zhao. Dynamic pricing in shared mobility on demand service. *arXiv preprint arXiv:1802.03559*, 2018.

[91] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.

[92] Panagiotis P Repoussis and Christos D Tarantilis. Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming. *Transportation Research Part C: Emerging Technologies*, 18(5):695–712, 2010.

[93] Stefan Ropke and David Pisinger. An adaptive large neighborhood search

heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[94] Guillaume Sabran, Samitha Samaranayake, and Alexandre Bayen. Precomputation techniques for the stochastic on-time arrival problem. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 138–146. Society for Industrial and Applied Mathematics, 2014.

[95] Samitha Samaranayake, Sebastien Blandin, and Alex Bayen. Speedup techniques for the stochastic on-time arrival problem. In *OASIcs-OpenAccess Series in Informatics*, volume 25. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[96] Samitha Samaranayake, Sebastien Blandin, and Alexandre Bayen. A tractable class of algorithms for reliable routing in stochastic networks. *Transportation Research Part C: Emerging Technologies*, 20(1):199–217, 2012.

[97] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[98] Martin WP Savelsbergh and Marc Sol. The general pickup and delivery problem. *Transportation science*, 29(1):17–29, 1995.

[99] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[100] Yu Shen, Hongmou Zhang, and Jinhua Zhao. Embedding autonomous vehicle sharing in public transit system: An example of last-mile problem. Technical report, 2017.

[101] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[102] Kevin Spieser, Samitha Samaranayake, Wolfgang Gruel, and Emilio Frazzoli. Shared-vehicle mobility-on-demand systems: A fleet operator's guide to rebalancing empty vehicles. In *Transportation Research Board 95th Annual Meeting*, number 16-5987, 2016.

[103] Kevin Spieser, Kyle Treleaven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in singapore. In *Road vehicle automation*, pages 229–245. Springer, 2014.

[104] Heinz Spiess and Michael Florian. Optimal strategies: a new assignment model for transit networks. *Transportation Research Part B: Methodological*, 23(2):83–102, 1989.

[105] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[106] Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael A Osborne. Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*, 2014.

[107] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[108] George Philip Szegö. *Towards global optimisation 2*, volume 2. North Holland, 1978.

[109] Tess Townsend. Google i/o 2017: Everything important that google announced today. `https://www.recode.net/2017/5/17/15654076/google-io-biggest-announcements-keynote-highlights-2017`, 2017.

[110] Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[111] Uber. How are fares calculated? `https://help.uber.com/h/33ed4293-383c-4d73-a610-d171d3aa5a78`, 2018.

[112] Akhil Vakayil, Wolfgang Gruel, and Samitha Samaranayake. Integrating shared-vehicle mobility-on-demand systems with public transit. Technical report, 2017.

[113] S Travis Waller and Athanasios K Ziliaskopoulos. On the online shortest

path problem with limited arc cost dependencies. *Networks*, 40(4):216–227, 2002.

[114] Ziyu Wang, Babak Shakibi, Lin Jin, and Nando Freitas. Bayesian multi-scale optimistic optimization. In *Artificial Intelligence and Statistics*, pages 1005–1014, 2014.

[115] Lixing Yang and Xuesong Zhou. Constraint reformulation and a lagrangian relaxation-based solution algorithm for a least expected time path problem. *Transportation Research Part B: Methodological*, 59:22–44, 2014.

[116] Lixing Yang and Xuesong Zhou. Optimizing on-time arrival probability and percentile travel time for elementary path finding in time-dependent transportation networks: Linear mixed integer programming reformulations. *Transportation Research Part B: Methodological*, 96:68–91, 2017.

[117] Dani Yogatama and Noah A Smith. Bayesian optimization of text representations. *arXiv preprint arXiv:1503.00693*, 2015.

[118] Liteng Zha, Yafeng Yin, and Zhengtian Xu. Geometric matching and spatial pricing in ride-sourcing markets. *Transportation Research Part C: Emerging Technologies*, 92:58–75, 2018.

[119] Jie Zhang, Ding Wen, and Shuai Zeng. A discounted trade reduction mechanism for dynamic ridesharing pricing. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1586–1595, 2016.

[120] Rick Zhang and Marco Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3):186–203, 2016.

[121] Wenwen Zhang and Subhrajit Guhathakurta. Parking spaces in the age of shared autonomous vehicles: How much parking will we need and where? *Transportation Research Record: Journal of the Transportation Research Board*, (2651):80–91, 2017.

[122] Wenwen Zhang, Subhrajit Guhathakurta, Jinqi Fang, and Ge Zhang. Exploring the impact of shared autonomous vehicles on urban parking demand: An agent-based simulation approach. *Sustainable Cities and Society*, 19:34–45, 2015.

[123] Yajie Zou, Xinxin Zhu, Yunlong Zhang, and Xiaosi Zeng. A space–time diurnal method for short-term freeway travel time prediction. *Transportation Research Part C: Emerging Technologies*, 43:33–49, 2014.