Tackling Car-Sharing Service Design Problems at Scale with High-Resolution Data: Discrete Simulation-Based Optimization Approaches

by

Tianli Zhou

B.Eng., Tsinghua University (2013)
S.M., Massachusetts Institute of Technology (2015)

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

Doctor of Philosophy in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author .....
Department of Civil and Environmental Engineering
March 13, 2020

Certified by .....
Carolina Osorio
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Colette L. Heald
Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee

Tackling Car-Sharing Service Design Problems at Scale with High-Resolution Data: Discrete Simulation-Based Optimization Approaches

by

Tianli Zhou

Submitted to the Department of Civil and Environmental Engineering
on March 13, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Transportation

Abstract

This thesis considers the design of two-way (i.e., round-trip) car-sharing services. The optimization problems are formulated as high-dimensional discrete simulation-based optimization (DSO) problems. Existing DSO algorithms cannot tackle these problems at scale. Moreover, they are designed based on asymptotic performance guarantees, but lack computational efficiency, i.e., they tend to not perform well under tight computational or simulation budgets. The main contribution of this thesis is to show how mixed-integer programming (MIP) models can be used to enable general-purpose DSO algorithms to become: (i) scalable: the car-sharing problems can now be tackled at scale; and (ii) computationally efficient: solutions with good performance can be identified given tight computational budgets. More generally, the methods proposed in this thesis contribute to bridging the gap between these two mostly disconnected research communities of analytical optimization and simulation-based optimization.

This thesis formulates MIP models and proposes two approaches to embed the MIP information within the DSO algorithms. First, we use a MIP to formulate a metamodel, which is an analytical approximation of the simulation-based objective function. The information from the MIP is used at every iteration of a DSO algorithm by solving an analytical metamodel optimization problem. Second, we use a MIP to enhance the partitioning step of an existing globally convergent DSO algorithm. The MIP is used to identify low-dimensional subregions of the feasible region, where more exhaustive simulation is to be carried out.

We then compare the performance of methods that either: (i) use the MIP information for metamodeling, (ii) use the MIP information for partitioning, or (iii) use the MIP information for both metamodeling and partitioning. We study how the MIP's accuracy impacts the performance of these methods.

Based on both small synthetic problems and a Boston area case study, we show how the scalability and the computational efficiency of both a general-purpose locally

convergent DSO algorithm and a general-purpose globally convergent DSO algorithm are enhanced. We also present results from a New York City case study. The case studies use detailed car-sharing reservation data from a major car-sharing operator. We benchmark the methods versus several algorithms, including stochastic programming. The combination of MIPs with DSO algorithms leads to methods with both asymptotic performance guarantees as well as good short-term performance.

Thesis Supervisor: Carolina Osorio
Title: Associate Professor of Civil and Environmental Engineering

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Professor Carolina Osorio, for everything you have done for me. Your guidance, expertise, patience and caring have supported my journey at MIT and contribute greatly to this dissertation. You show me how to be a good researcher. It is my honor to know you.

I would also thank Professor Patrick Jaillet and Professor Saurabh Amin for being my dissertation committee members. Your suggestions on both dissertation contents and presentation skills are very valuable. I have learned a lot from you.

I would like to thank Kiley Clapper, Max Martelli, Sarah Smith, and Roberta Pizzinato of CEE for all their administrative support during past years. I would also like to appreciate the Ford-MIT Alliance and the National Science Foundation for sponsoring my study. I thank Zipcar for providing the data which has made my research possible.

I would like to thank Chancellor Cynthia Barnhart, who was my Master's advisor at MIT, and Professor Hai Jiang, who taught me in Tsinghua University. You opened the door of optimization and transportation to me.

Many thanks go to my Osorio Lab members: Linsen Chong, Jing Lu, Timothy Tay, Chao Zhang, Kevin Zhang, and Nate Bailey. Special thanks go to Evan Fields, for our collaboration on the Zipcar project. My labmates have made my study at MIT enjoyable and unforgettable. And I also want to thank all my CEE fellow students who make our community diverse and vibrant.

Additionally, many thanks go to my friends Manxi Wu, Yin Wang, Yan Zhao, Xiao Fang, Haizheng Zhang, Hongyi Zhang, Li Jin, Hai Wang, Joanna Moody, Chiwei Yan, Yiwen Zhu, Zhan Zhao, Yan Leng, Keji Wei, Manish Singh, Yoo Joon Kim, and Seonkyoo Yoon. You are like brothers and sisters to me. Your sincere care and suggestions helped me survive graduate school. You are incredible.

Finally, I would like to thank my father Xiaorui Zhou, my mother Yueming Liu and my fiancée Xiaoying Yu. Your love and support are, and will always be, my strongest power to move forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Dissertation motivation and objective

In recent years, the most successful trend in the space of urban mobility services has been the widespread use of shared mobility services, such as ride-sharing, car-sharing, bike-sharing and, most recently, scooter-sharing (Shaheen and Chan, 2016). Major technology companies have been behind the rapid growth of these shared services. The operators of these services collect abundant data of the usage of the vehicles and the behavior of the clients (or users). This data provides a high-resolution disaggregate description of the interaction of demand and supply. This work is motivated by the following research question: how can we exploit the rich disaggregate information in this data to optimize the design and the operations of these new urban mobility services?

The most common approach to address these optimization problems is to aggregate the data such as to estimate parameters of a mathematical program, such as a mixed-integer programming model (MIP) or a stochastic programming model (SP). These mathematical programs provide an aggregate description of both demand and of the interaction of demand and supply. This aggregate description enables their computational tractability and their scalability (i.e., their use for large-scale instances). Nonetheless, through this aggregation a wealth of information of the intricate interactions between demand and supply is lost. Losing such information may lead to a less

detailed understanding of the mobility-sharing system. Chen et al. (2019b) found that aggregated data would disregard individual heterogeneity, and hence underestimate level of unevenness of the distribution of the accessibility of a bike-sharing system.

A current trend among major technology companies is to design optimization methods that exploit the rich information in their disaggregate data. Companies such as Lyft are building high-resolution simulators of their services that sample directly from their disaggregate data and provide a disaggregate description of the performance of their service (Greenhall, 2016). Hence, the next generation of mobility optimization algorithms will increasingly perform optimization based on models that provide a disaggregate description of mobility.

This work addresses this need. It formulates a car-sharing optimization problem as a simulation-based optimization (SO) problem, and proposes computationally efficient SO algorithms. We use a disaggregate car-sharing service simulator, which was developed in collaboration with Ford and with the car-sharing operator Zipcar (Fields et al., 2017). The simulator samples from disaggregate car-sharing reservation data to estimate (disaggregate) demand (i.e., it yields a set of desired reservations) and then provides a simple stochastic mapping of how this demand interacts with supply to yield disaggregate reservations (i.e., a final set of realized reservations).

There are challenges remaining for applying the existing SO algorithms to address this high-dimensional car-sharing optimization problem. Most discrete SO algorithms are general-purposes and do not account for problem specific information. Due to the curse-of-dimensionality problem of discrete optimization, most of them do not scale. Problems with more than 10 decision variables are considered as high-dimensional in the context of discrete SO (Xu et al., 2013). These algorithms usually have local or global convergence guarantees, but are not designed to have good finite time performance.

To address these scalability and computational efficiency challenges, we propose to formulate and embed analytical mathematical optimization models (more specifically, mixed-integer formulations) within the discrete SO algorithms. The proposed approaches maintain the local or global convergence guarantees of the underlying

discrete SO algorithms and enable these algorithms to become scalable and computationally efficient. The proposed algorithms and case studies illustrate how abundant disaggregate mobility data can be used to perform large-scale (e.g., city-scale) optimization.

1.2 Dissertation structure and contributions

This dissertation contains four chapters. In Chapter 2, we present MetaAHA, an efficient locally convergent discrete SO algorithm. The contributions of this chapter can be summarized as follows.

- **Data-driven technique** The most traditional approach to car-sharing service optimization has been analytical optimization. This comes at the cost of a simplified description of demand and of demand-supply interactions. In this work, our goal is to acknowledge both the intricacy of a car sharing service (e.g., intricate demand distribution, intricate demand-supply interactions), as well as the availability of high-resolution data. Hence, we propose a method that relies heavily on the rich reservation data and uses limited modeling assumptions. The information captured in the data about the underlying demand distribution and demand-supply interactions is preserved and exploited at a disaggregate level. To the best of our knowledge, this is the first work to design algorithms that preserves this high-resolution information of the data (i.e., does not merely aggregate the disaggregate data) for car-sharing network design optimization. Case studies with data from Zipcar's Boston market are carried out.

- **High dimensional discrete SO problems** The proposed algorithm is suitable to address high-dimensional network design problems. In Section 2.3.3 and 2.3.4, we use it to address a Boston metropolitan area case study with 315 stations. General-purpose discrete SO algorithms have been extensively used to tackle problems with roughly 20 decision variables. Our enhanced scalability comes at the cost of proposing an algorithm tailored for a specific class of

network design problems, while the general-purpose algorithms can be used for a broader class of general discrete SO problems. We achieve scalability by formulating and embedding information from a MIP within the general-purpose algorithm AHA proposed by Xu et al. (2013). This yields the proposed discrete SO algorithm, which we call MetaAHA. The approach combines the merits of both analytical optimization methods (i.e., tractability and scalability) and of simulation-based optimization methods (i.e., we can sample directly from the disaggregate data to enable a detailed description of demand and of demand-supply interactions).

- **Computationally efficient algorithms** The proposed algorithms are designed to identify good quality solutions within few iterations (i.e., when few simulation observations are available). This differs from most discrete SO literature which is focused on asymptotic performance. This efficiency is achieved through the novel metamodel formulation which embeds a non-simulation-based representation (a MIP formulation) of the network design problem. In other words, the simulator is no longer treated as a black box, instead analytical problem-specific information is embedded within the SO algorithm. We propose both locally convergent and globally convergent frameworks. The results of Section 2.3 indicate that this analytical structural information is the key to achieving computational efficiency. They also illustrated how the combination of the proposed metamodel along with a general-purpose discrete SO algorithm yields an algorithm with both good short-term and asymptotic performance properties. Moreover, the metamodel enables the general-purpose algorithm to become robust to the quality of the initial solution.

- **Metamodeling for discrete SO** The main feature of the proposed algorithm is the formulation of a metamodel, (i.e., an analytical approximation of the simulation-based objective function) that has a functional form that is problem-specific. Such metamodel ideas for transportation problems have been successfully formulated for various continuous SO problems. This is the first work

that extends these ideas to the discrete SO setting. This work shows that by using such metamodel ideas, high-dimensional discrete SO problems can be addressed in a computationally efficient way. The chapter shows how the proposed metamodel ideas enable general-purpose discrete SO algorithms to become more scalable (i.e., suitable for higher-dimensional problems). Since fundamental OR transportation optimization problems (e.g., routing) are naturally formulated as discrete optimization problems, the ideas of this work lay the foundations for a variety of important and difficult transportation problems to be addressed efficiently with data-driven, or simulation-based, network models.

In Chapter 3, we present MetaESBB-OptDim, a globally convergent discrete SO algorithm. The contributions of this chapter are summarized as the follows.

- **Computationally efficient globally convergent discrete SO algorithm** We extend an existing globally convergent discrete SO algorithm, and enable it to become efficient and scalable. We achieve this by formulating a MIP and combining information from the simulator with information from the MIP. The proposed algorithm can identify solutions with good performance in only a few iterations.

- **Enhanced computational efficiency** We use the information from the MIP in two ways: (1) we use the MIP to formulate a metamodel and solve a metamodel optimization problem (this is in line with the method proposed in Chapter 2, and (2) we use the MIP to partition the feasible region, i.e., we use it to identify a low-dimesional subregion of the feasible region where more extensive simulation is to be carried out. As far as we know, this is the first method in discrete SO that use problem specific information to guide the partitioning procedure.

In Chapter 4, we conclude this work and discuss future research directions.

1.3 Bibliographic notes

Some contents of this dissertation (mainly Chapter 2) stem from the following write-up:

- Tianli Zhou, Carolina Osorio, and Evan Fields. A data-driven discrete simulation-based optimization algorithm for car-sharing service design. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2019. Available at: `http://web.mit.edu/osorioc/www/papers/zhoOsoFieCarSharing.pdf`. (Submitted for publication)

Chapter 2 of this dissertation has been presented in the following conference:

- Tianli Zhou, Carolina Osorio, and Evan Fields. "A Data-Driven Discrete Simulation-Based Optimization Algorithm for Large-Scale Two-Way Car-Sharing Network Design", INFORMS Transportation Science and Logistics (TSL) Society First Triennial Conference, Chicago, Illinois, USA (Jul 2017).

Chapter 3 of this dissertation has been presented at the following conferences:

- Tianli Zhou, Carolina Osorio. "A Globally Convergent Discrete Simulation-Based Optimization Algorithm for Complementing Transit Accessibility with Car-Sharing Service". 7th INFORMS TSL Society Workshop, Vienna, Austria (Jul 2019).

- Tianli Zhou, Carolina Osorio. "Car-sharing Service Design: A Simulation-based Optimization Approach For High Dimensional Problem". 2019 INFORMS Annual Meeting, Seattle, Washington, USA (Oct 2019).

Chapter 2

A a locally convergent discrete SO algorithm suitable for high-dimensional car-sharing service design problems

2.1 Introduction

In this chapter, we propose a discrete simulation-based optimization (SO) algorithm for a family of large-scale car-sharing network design problems. We focus on two-way car-sharing services, and aim to assign a fleet to a network of car-sharing stations to maximize the profit of the service. The proposed approach is a metamodel SO approach. The metamodel in the form of mixed-integer program (MIP) is formulated. The metamodel is embedded within Adaptive Hyperbox Algorithm (AHA), a general-purpose locally convergent discrete SO algorithm. The proposed algorithm is validated with synthetic toy network experiments. The algorithm is then applied to a high-dimensional Boston case study using reservation data from Zipcar, a major US car-sharing operator. The method is benchmarked versus several algorithms, including stochastic programming. The experiments indicate that the analytical network

model information, provided by the MIP to the SO algorithm, is useful both at the first iteration of the algorithm and across subsequent iterations. The solutions derived by the proposed method are benchmarked versus the solution deployed in the field by the car-sharing operator.

2.1.1 Car-sharing service optimization

Car-sharing has become a popular transportation mode in urban areas in the past decades. Its deployment, as of 2010, covered over 31,600 vehicles in over 1,100 cities in 26 countries with over 1 million members (Shaheen and Cohen, 2013). The car-sharing literature has studied its potential to reduce the transportation cost of households (Duncan, 2011), to complement private-vehicle ownership (Shaheen and Cohen, 2013, Becker et al., 2017) and public transportation systems (Chiraphadhanakul 2013 Chapter 4, Nair and Miller-Hooks 2014, Zhou 2015 Chapter 3), as well as to mitigate greenhouse gas emissions and total vehicle miles traveled (Firnkorn and Müller, 2011, Shaheen and Cohen, 2013).

The main types of car-sharing service are two-way (also known as round-trip), one-way station-based, free-floating and peer-to-peer. For full definitions, see for instance Schmöller et al. (2015). A station is a location with a certain number of vehicle-sharing parking spots. Two-way services consist of a set of vehicles parked at a set of fixed stations. In advance, customers reserve a vehicle for a given duration and a given start time. They then pick-up and drop-off the vehicle from the same predetermined station. Reservations can be made from several months in advance to minutes in advance. There is no upper limit on the duration of a reservation. As of July 2015, there were an estimated 1.17 million two-way service members along with 0.31 million one-way service members in the United States (NCSL, 2020). This chapter focuses on two-way services with an application to a Boston case study with Zipcar data. Zipcar is a major car-sharing service provider in the US. It is also one of the world's largest car-sharing service provider with operations in more than 500 cities worldwide. It has deployed over 12,000 vehicles around the world (Zipcar, 2017). Currently, Zipcar offers two-way service, one-way station-based service and

free-floating service. Round-trip is the primary service mode for Zipcar and the foundation of its business. Studying the optimization of its two-way service is critical for the business of Zipcar's business.

The data we use in this chapter consists of two-way car-sharing reservations made in the Boston metropolitan area. Each reservation contains detailed information, such as creation time (time at which the reservation was made), duration, start time, end time, station (location where the vehicle is to be picked up and dropped off), and other vehicle attributes. Hereafter, we use the term data-driven to emphasize that: (i) unlike most approaches, we do not aggregate the data, instead we sample directly from the disaggregate reservation data to yield a disaggregate description of latent demand; (ii) we then use the disaggregate latent demand as input to a (disaggregate) simulator that mimics the reservation process or behavior of individual clients (e.g., if their desired reservation is not available, they may consider opting out or opting for a reservation at another station or time); this yields a disaggregate set of realized (through simulation) reservations. The optimization problem studied in this chapter is the optimal spatial allocation of a fleet of two-way car-sharing vehicles to a set of stations. This is a tactical decision that car-sharing operators typically make on a monthly basis. The corresponding optimization problem is solved offline.

Detailed reviews of vehicle-sharing studies are given in Jorge and Correia (2013), Brandstätter et al. (2016). Table 2.1 summarizes some of the recent vehicle-sharing network design literature. The column "Optimization" indicates whether the method is analytical, simulation-based or a combination of both. The column "Context" specifies the type of vehicle-sharing service (one-way, two-way, free-floating) and the type of vehicle (bike, car). The column "Case study size" indicates, for the main case study of each paper, the number of sites (e.g., locations, regions, stations), the number of integer and the number of continuous variables (including both decision variables and auxiliary variables). Cells are left blank for cases where these numbers are not directly reported in the papers. The "Problem" column specifies the type of decisions the problem addresses.

The most popular approach to address vehicle-sharing (both car- and bike-sharing)

Table 2.1: Summary of recent related vehicle-sharing network design papers

Paper	Optimization		Context					Case study size			Problem			
	Analytical	Simulation-based	One-way	Two-way	Free-floating	Bike-sharing	Car-sharing	Site	Integer	Continuous	Site location	Fleet assignment	Station capacity	Other
Correia and Antunes (2012)	✓		✓				✓	75		0	✓	✓	✓	Rebalance fleet
Cepolina and Farina (2012)		✓	✓				✓	11	9	0			✓	Fleet size
Chiraphadhanakul (2013, Chapter 4)	✓		✓			✓		27	27		✓			Route user flow
Correia et al. (2014)	✓		✓				✓	116		0	✓	✓		Select trips
Nair and Miller-Hooks (2014)	✓		✓			✓	✓	64	4420	6295	✓	✓	✓	Route user flow
Boyacıet al. (2015)	✓		✓				✓	100	All together $\sim 10^5$		✓		✓	Determine fleet size, regions served by each station and number of relocation personnel, rebalance fleet
Deng (2015, Chapter 5)		✓	✓				✓	8	11	2		✓	✓	Determine fleet size, rebalance fleet
Jorge et al. (2015)	✓		✓	✓			✓	391		0		✓	✓	Select trips, rebalance fleet
O'Mahony (2015, Chapter 3)	✓		✓			✓		300				✓	✓	
Zhou (2015, Chapter 4)	✓		✓			✓		30	5133	$\sim 1.5 \times 10^7$	✓			Route user flow
Jian et al. (2016)	✓	✓	✓			✓		466	932	0		✓	✓	
Lu et al. (2017)	✓		✓	✓	✓		✓	9				✓	✓	Route user flow, rebalance fleet
He et al. (2017)	✓				✓		✓	61			✓			Route user flow, rebalance fleet
This thesis	✓	✓		✓			✓	315	315	0		✓		

network design problems across all service types (two-way, one-way, floating) is the use of analytical mixed integer programming (MIP). Studies with deterministic demand include Correia and Antunes (2012), Chiraphadhanakul (2013, Chapter 4), Correia et al. (2014), Nair and Miller-Hooks (2014), Zhou (2015, Chapter 3). Past work in the field has also accounted for demand uncertainty by using a parametric probability distribution for demand combined with optimization methods such as stochastic programming and robust optimization (O'Mahony 2015, Chapter 4, Lu et al. 2017, He et al., 2017).

Car-sharing demand-supply interactions are intricate to model, yet are critical to account for when planning and operating car-sharing services. Studies of car-sharing demand include Millard-Ball et al. (2005), Stillwater et al. (2009), Ciari et al. (2013), De Lorimier and El-Geneidy (2013), Coll et al. (2014), Ciari et al. (2014, 2016b). The analytical modeling of demand involves accounting for how the distribution of demand varies as a function of space, time, user-specific attributes (e.g., value of time, willingness to walk, trip purpose) and other transportation system attributes (e.g., alternative travel modes for that user and that trip purpose). Moreover, for two-way car-sharing, the analytical modeling of the interaction of demand and supply is particularly difficult due to the often low supply capacity: there are typically few car-sharing parking spots available at each station. Hence, if a user does not find a vehicle available at the desired time and station, the user may opt out of renting a vehicle (the demand is said to be lost, and the historical reservation data is aid to be truncated) or may opt into renting a nearby (e.g., in space, in time) reservation (the demand is said to spillover or spillback and the historical reservation is said to be censored). For a detailed description of truncation and censoring in the context of car-sharing, see Fields et al. (2018). The likelihood of truncation and of censoring can depend on user characteristics (e.g., willingness to walk, car ownership), on trip attributes (e.g., trip purpose) as well as on the general mobility system (e.g., availability of other competitive travel alternatives). Additionally, given this low supply capacity, it is important to account for the temporal order in which users make reservations. In other words, modeling the *first-come-first-reserve* principle (i.e., the

fact that reservations are prioritized or processed in the order of their creation time) is important. Due to the difficulties of accurately modeling car-sharing demand, as well as demand-supply interactions, we propose to directly use disaggregate car-sharing reservation data that embeds a detailed description of the interaction of demand and supply.

Compared to pure analytical models, stochastic simulators enable a more detailed modeling of demand and supply uncertainties, and of demand-supply interactions, their use to address optimization problems of realistic dimensions remains intricate. In the context of vehicle-sharing, simulation tools have mostly been used to evaluate the performance of network designs obtained from analytical models, i.e., the simulator is used to perform what-if analysis (Cepolina and Farina 2012, O'Mahony 2015 Chapter 5, Ciari et al. 2015). Various simulation studies that account for car-sharing (Ciari et al., 2009, 2016a, Balać et al., 2016, Balac et al., 2017) have been carried out with the MATSim transportation simulation software (MATSim, 2018). Studies, such as Cepolina and Farina (2012) and Deng (2015, Chapter 5), have included the simulator as part of an optimization framework and have resorted to general-purpose black-box optimization algorithms such as simulated annealing and particle swarm optimization. The study of Jian et al. (2016) exploited problem-specific information to yield gradient-type information. Interestingly, Jian et al. (2016) use the solution of an analytical linear integer program as the initial solution for a simulation-based optimization algorithm. Such an approach is also used as benchmark method in the case studies of this chapter. Of particular notice is the large-scale bike-sharing optimization instance studied in Jian et al. (2016), which considers a set of 466 stations.

2.1.2 Discrete simulation-based optimization

We formally define the discrete SO problem. Let $\boldsymbol{x} \in \mathcal{X} \cap \mathbb{Z}^I$ be our decision vector of dimension I, where $\mathcal{X} \subset \mathbb{R}^I$ is convex and bounded. For any $\boldsymbol{x} \in \mathcal{X} \cap \mathbb{Z}^I$, $G(\boldsymbol{x})$ is a random variable. We can observe the value of $G(\boldsymbol{x})$ via stochastic simulation. Our

goal is to solve the optimization problem:

$$\max_{\boldsymbol{x}\in\mathcal{X}\cap\mathbb{Z}^I} g(\boldsymbol{x}) = E[G(\boldsymbol{x})]. \tag{2.1}$$

In this chapter, in order to enable the direct use of disaggregate car-sharing reservation data for optimization, we formulate the problem as a discrete simulation-based optimization (SO) problem. The problem has a simulation-based objective function with discrete decision variables. Constraints are analytical (i.e., they are not simulation-based). The main challenges of addressing such problems are the following. There is no analytical expression available for the objective function, hence traditional (analytical) discrete optimization algorithms cannot be used. The objective function can only be estimated by running a set of stochastic simulation replications. Discrete SO problems inherit the curse of dimensionality of discrete analytical problems. Since simulation is used, the objective function is often an intricate (e.g., non-convex) function of the decision variables with several local optima.

There are a variety of discrete SO algorithms in the literature; recent reviews include Nelson (2010) and Hong et al. (2015). Discrete SO algorithms include Convergent Optimization via Most-Promising-Area Stochastic Search (COMPASS) (Hong and Nelson, 2006), Adaptive Hyperbox Algorithm (AHA) (Xu et al., 2013), R-SPLINE (Wang et al., 2013), and cgR-SPLINE (Nagaraj, 2014). Methods that aim to identify solutions with good performance at an early stage (i.e., within few simulations) include an extension of COMPASS known as the Industrial Strength COMPASS (ISC) (Xu et al., 2010), as well as extension of AHA known as ISC-AHA (Xu et al., 2013). Other common approaches to discrete SO problems include ranking-and-selection (R&S) techniques, such as Chick and Inoue (2001), Frazier et al. (2008). An R&S review can be found in Swisher et al. (2003).

Discrete SO algorithms are most often designed: (i) as general-purpose algorithms, i.e., they can be used to address a broad family of optimization problems, their use is not limited to transportation problems, and (ii) based on asymptotic convergence properties, there is limited focus on their short-term (i.e., small sample performance).

The performance of these general-purpose discrete SO algorithms is typically illustrated with low-dimensional problems (e.g., around 20 decision variables). Few studies have reported higher-dimensional instances. The work of Xu et al. (2013) reported experiments where AHA successfully addressed problems with up to 100 decision variables. Developing discrete SO algorithms that are suitable for high-dimensional problems remains a challenge. Past studies, such as Xu et al. (2013), illustrate that for locally convergent general-purpose discrete SO algorithms, the quality of the final solution is sensitive to the quality of the initial solution. Hence, there is also an interest to develop algorithms with enhanced robustness to the quality of the initial solution.

There is a lack of studies that evaluate the performance of general-purpose discrete SO algorithms for high-dimensional problems and under tight computational budgets (i.e., within few simulation runs). Nonetheless, when using simulators to address optimization problems, practitioners often use the algorithms under tight computational budgets (e.g., they terminate the algorithm once a fixed time or a fixed number of iterations have elapsed). Hence, there is a need for computationally efficient algorithms. These are algorithms that provide solutions with improved performance (compared to initial solutions or solutions deployed in the field) within few simulation runs.

In transportation, fundamental optimization problems are naturally formulated as discrete problems. Additionally, realistic case studies quickly lead to high-dimensional instances. Hence, this chapter designs a discrete SO algorithm that is both computationally efficient and suitable for high-dimensional problems. Moreover, the case studies of this chapter, illustrate the robustness of the algorithm to the quality of the initial point.

This chapter focuses on metamodel SO approaches. In past work, we have formulated metamodel SO algorithms for various continuous SO transportation problems (Osorio and Nanduri, 2015, Chong and Osorio, 2017, Zhang et al., 2017, Chen et al., 2019a, Osorio, 2019). A recent review of metamodel SO methods appears in Osorio and Chong (2015). A more detailed description of commonly used metamodels is

given in Section 2.2.2. To the best of our knowledge, the use of metamodel approaches for discrete SO has been limited to low-dimensional problems (with up to 15 decision variables). In the broader area of transportation (i.e., not limited to vehicle-sharing) discrete SO has been used in studies such as Jung et al. (2014), Chen et al. (2015), Sebastiani et al. (2016), Jian et al. (2016), Boyacıet al. (2017).

This chapter formulates a novel metamodel for a family of car-sharing SO problems. We then combine the metamodel with an existing general-purpose discrete SO algorithm, known as AHA, leading to a novel metamodel SO algorithm. The proposed algorithm preserves the asymptotic convergence properties of the general-purpose algorithm. More specifically, the proposed algorithm remains a locally convergent algorithm.

In this chapter, we use a car-sharing network simulator (Fields et al., 2017), which relies on few demand modeling assumptions. Instead, it relies primary on sampling from disaggregate car-sharing reservation data. It provides a detailed description of the spatial-temporal distribution of demand as well as of demand-supply interactions. Unlike most methods that aggregate the data to fit aggregate model parameters, we use the data in disaggregate form. Hence, we refer to our method as a data-driven SO algorithm.

In the following parts of this chapter, Section 2.2 formulates the proposed methodology. Its performance is evaluated and benchmarked in Section 2.3 with experiments on both synthetic toy networks and Boston networks. Summary of this chapter are presented in Section 2.4. Algorithmic details are presented in Appendix B. The formulation of the SP model that is used as a benchmark in Section 2.3.4 is given in Appendix C. Additional implementation detailes are given in Appendix D.

2.2 Methodology

This section presents the proposed methodology. The network design problem is formulated in Section 2.2.1. The general metamodel SO framework is discussed in Section 2.2.2. The metamodel for the considered car-sharing network design problem

is formulated in Section 2.2.3 and the proposed algorithm is described in Section 2.2.4. The car-sharing network simulator used in this chapter as well as the role of the car-sharing data are summarized in Section 2.2.5.

2.2.1 Network design problem formulation

We consider a two-way car-sharing system from the perspective of the car-sharing operator. The network design problem is to assign a fleet of vehicles across a network of stations such as to maximize the expected profit. We also refer to this problem as the fleet assignment problem. The network design problem is studied for a given finite time horizon, which we refer to as the planning period. To formulate the problem, we introduce the following notation.

- x_i: number of cars assigned to station i (decision variable);
- $\boldsymbol{x}$: vector of x_i's for all $i \in \mathcal{I}$;
- $R(\boldsymbol{x}; \boldsymbol{q}_1)$: random variable representing the revenue;
- $g(\boldsymbol{x}; \boldsymbol{q}_1)$: expected profit (SO objective function);
- c_i: cost, over the planning period, of a parking space at station i;
- $\boldsymbol{q}_1$: exogenous simulation parameter vector (e.g., reservation pricing);
- N^i: capacity of station i (i.e., number of parking spots);
- X: total fleet size (i.e., number of cars to assign);
- I: total number of stations;
- $\mathcal{I}$: set of all stations, $\mathcal{I} = \{1, 2, \ldots, I\}$;
- $\mathcal{F}$: feasible region.

The problem is formulated as follows:

$$\max_{\boldsymbol{x}} \quad g(\boldsymbol{x}; \boldsymbol{q}_1) = E[R(\boldsymbol{x}; \boldsymbol{q}_1)] - \sum_{i \in \mathcal{I}} c_i x_i \tag{2.2}$$

subject to

$$\sum_{i \in \mathcal{I}} x_i \leq X \tag{2.3}$$

$$x_i \leq N^i \quad \forall i \in \mathcal{I} \tag{2.4}$$

$$x_i \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}. \tag{2.5}$$

The objective function represents the expected profit for a given fleet assignment vector, $\boldsymbol{x}$. It is defined as the difference between the expected revenue $E[R(\boldsymbol{x}; \boldsymbol{q}_1)]$ and the costs. The expected revenue is a simulation-based function, estimates of which can be obtained via simulation. The cost parameters, c_i, are exogenous. In this work, they represent parking space leasing fees. Constraint (2.3) bounds the total number of cars assigned across all stations with the fleet size. Constraint (2.4) bounds the number of cars assigned to each station i with the space capacity of the station. The number of cars assigned to each station are assumed to be non-negative integers (Constraint (2.5)). Constraints (2.3)-(2.5) specify the feasible region, $\mathcal{F}$.

The expectation in the objective function accounts for the stochasticity in the simulation process. The simulator, which is summarized in Figure 2-1 and described in more detail in Section 2.2.5, combines a sampling procedure that samples from a set of car-sharing reservation data and an assigning procedure that determines whether a reservation request will be satisfied and how it will be satisfied. In other words, realizations of the revenue random variable R are obtained by sampling from car-sharing reservation data. The sources of uncertainty include: (i) a stochastic sampling process that samples from the historical reservation data to infer a set of disaggregate latent demand (or desired reservations); (ii) a stochastic description of how demand and supply interact and how truncation and censoring occur (e.g., probability with which a user for which his/her desired reservation is not available, decides to opt out of making a reservation or decides to find a substitute reservation).

The challenges of addressing discrete SO problems, such as Problem (2.2)-(2.5), were detailed in Section 2.1. Given these challenges, we propose an algorithm that at every iteration, uses the set of estimates of g obtained so far to formulate and solve an

(approximate) analytical discrete problem that: (i) provides good quality solutions to the underlying SO problem, (ii) can be solved efficiently for high-dimensional instances, and (iii) can be solved with a variety of widely-used commercial solvers.

Note that the traditional approach to address this fleet assignment problem is to formulate it as a MIP (such a formulation is given in Section 2.2.3), or as a stochastic programming model. The problem would no longer be simulation-based, and hence large-scale instances could be solved efficiently. Nonetheless, this would come at the cost of using only aggregate information from the car-sharing reservation data (since the data would merely be used to fit a set parameters of the analytical mathematical programs) and of embedding a simplified description of demand-supply interactions, as is detailed below.

A stochastic programming model is used as a benchmark method in the case study of Section 2.3.4. Compared to its SP counterpart, the simulator provides a more detailed description of the truncation and of the censoring of demand. First, it satisfies the first-come-first-reserve principle, i.e., the desired reservations are processed in the order of their creation time (which is obtained from the historical data). Second, when a user's desired reservation is not available and he/she decides to consider alternate substitute reservations, the simulator considers a sequential process where feasible substitute reservations are ranked by a distance metric (for instance, available vehicles that are closer in spatial or temporal distance, are more likely to be a substitute than those further away). These aspects could be naturally formulated as analytical nonlinear functions, or based on analytical linear approximations. Nonetheless, the resulting SP would be less tractable and less scalable.

As is detailed below, the proposed methodology combines the advantages of using a simulation-based model (which allows for the use of more detailed models and historical data) and those of an analytical mathematical program (which allows for computational tractability and scalability).

2.2.2 General metamodel approach

Let us first briefly present the main ideas of the metamodel SO approach, which are based on the continuous SO framework of Osorio and Bierlaire (2013). To formulate the problem, we introduce the following notation.

- k: iteration index of the SO algorithm;
- m_k: metamodel at SO iteration k;
- $\boldsymbol{\beta}_k$: vector of metamodel parameters at SO iteration k, element i is denoted $\beta_{k,i}$;
- $\boldsymbol{z}$: vector of endogenous variables;
- $\boldsymbol{q}_2$: vector of exogenous parameters;
- g_A: approximation of g (Equation (2.2)) derived by the analytical;
- h: constraints of the analytical network model.

The main idea of metamodel SO is to replace the simulation-based objective function (2.2) with an analytical approximation, which is known as the metamodel. In the metamodel literature, general-purpose functions (e.g., low-order polynomial functions, radial-basis functions, Kriging functions) are the most common choice both for continuous SO problems (Jones et al., 1998, Barton and Meckesheimer, 2006, Wild et al., 2008, Kleijnen et al., 2010, Ankenman et al., 2010) and for discrete SO problems (Xu 2012, Sun et al. 2014, Salemi 2014 Chapter 4, Xie et al. 2016). They are chosen based on their mathematical properties. They are referred to as general-purpose functions because their choice does not depend on the specific problem formulation (i.e., their functional form is invariant to the choice of the objective function (2.2)). Nonetheless, due to this generality, their functional form does not embed any problem-specific structural information. Osorio and Bierlaire (2013) propose to formulate metamodels that embed problem-specific information. By doing so,

the resulting SO algorithms have enhanced computational efficiency, scalability and robustness to both simulator stochasticity and to the quality of the initial solutions.

In this chapter, we follow the idea of Osorio and Bierlaire (2013). The metamodel is defined by (2.6) as the sum of a problem-specific function (g_A) and a general-purpose linear function (term within parenthesis of (2.6)). The problem-specific function (g_A) is the analytical objective function of a mathematical program (more specifically of a mixed-integer linear fleet assignment problem), which embeds a simplified representation (compared to the simulator) of the mapping between the supply configuration ($\boldsymbol{x}$) and the expected profit (g of (2.2)). The goal of g_A is to provide a good analytical approximation of the simulation-based objective function for the considered problem. Nonetheless, this analytical approximation is not expected to be accurate (due to the more detailed and intricate models of demand and of supply embedded in the simulator, that are not accounted for in the mathematical program). Hence, the metamodel (2.6) can be thought of as the objective function of a MIP that is corrected for parametrically by both a scaling term (scalar $\beta_{k,0}$ of (2.6)) and an additive linear error term (term within parenthesis of (2.6)). To the best of our knowledge, this is the first work to consider a metamodel that combines both a problem-specific component and a general-purpose component for discrete SO problems.

At a given iteration k of the SO algorithm, we solve the following analytical problem, referred to as the metamodel optimization problem.

$$\max_{\boldsymbol{x},\boldsymbol{z}} \quad m_k(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\beta}_k,\boldsymbol{q}_2) = \beta_{k,0} g_A(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2) + \left(\beta_{k,1} + \sum_{i\in\mathcal{I}} \beta_{k,i+1} x_i\right) \tag{2.6}$$

$$h(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2) = 0 \tag{2.7}$$

$$\boldsymbol{x} \in \mathcal{F}. \tag{2.8}$$

Since g_A is the objective function of a MIP, the corresponding constraints of the MIP are represented here through the function h of (2.7). These are formulated in detail in Section 2.2.3. The constraints of Section 2.2.3 consist of both equality and inequality constraints. They are summarized here as a set of equality constraints (they can equivalently be represented as a set of inequality constraints).

The metamodel optimization Problem (2.6)-(2.8) differs from the simulation-based optimization Problem (2.2)-(2.5) in that: (i) it replaces the (unknown) simulation-based objective function (g of (2.2)) with an analytical function (m_k of (2.6)); (ii) it has additional constraints (Eq. (2.7)). The main feature that has allowed us in the past to design efficient algorithms for continuous SO problems is the formulation of a metamodel that embeds an analytical and problem-specific approximation of $g(\boldsymbol{x})$. This is the key component of the approach, yet this is also where the main methodological challenge lies because it is necessary to formulate an analytical model that: (i) provides a good approximation of the intricate function $g(\boldsymbol{x})$, which as will be discussed in Section 2.2.3 is particularly difficult for this car-sharing context, (ii) is scalable (i.e., is suitable to address high-dimensional instances), and (iii) is computationally efficient. The latter is critical because the metamodel optimization problem is solved at *every* iteration of the SO algorithm. Hence, it should be sufficiently efficient to warrant the allocation of computing resources to solving it rather than to running the simulator (i.e., simulating new points or increasing the accuracy of the estimates of simulated points).

The metamodel (m_k of (2.6)) is a parametric function with parameter vector $\boldsymbol{\beta}_k$. The latter are fitted, at every iteration of the SO algorithm, by solving a problem that minimizes a least squares distance between metamodel predictions and simulation observations. For more details, see Problem (A.1) in Appendix A. As discussed above, the main challenge in this approach is the formulation of a computationally efficient and scalable problem-specific approximation of g, denoted here g_A. Let us now present the proposed formulation.

In Figure 2-1, we show the basic logic of the data-driven metamodel SO framework. The main idea is that historical disaggregate car-sharing reservation data (which represents potentially censored or truncated demand) is used to estimate disaggregate latent (i.e., uncensored and untruncated) demand (for a detailed description of this demand estimation methodology, see Fields et al. (2018)). For a given latent demand and a given supply solution (i.e., a given spatial allocation of the vehicle fleet), the simulator stochastically evaluates the performance of the solution, this yields a set of

disaggregate reservations (which may have been censored).

Every time new supply solutions are evaluated via simulation, the metamodel is updated (i.e., its parameters are fitted by using the current set of simulation observations) and then used to solve an analytical optimization problem. More specifically, the metamodel optimization problem is a mixed-integer program (MIP). While the analytical metamodel (i.e., the MIP) provides an aggregate description of demand and of demand-supply interactions, the simulator operates based on a disaggregate representation of demand (i.e., individual desired reservations) and of demand-supply interactions (i.e., individual realized reservations).

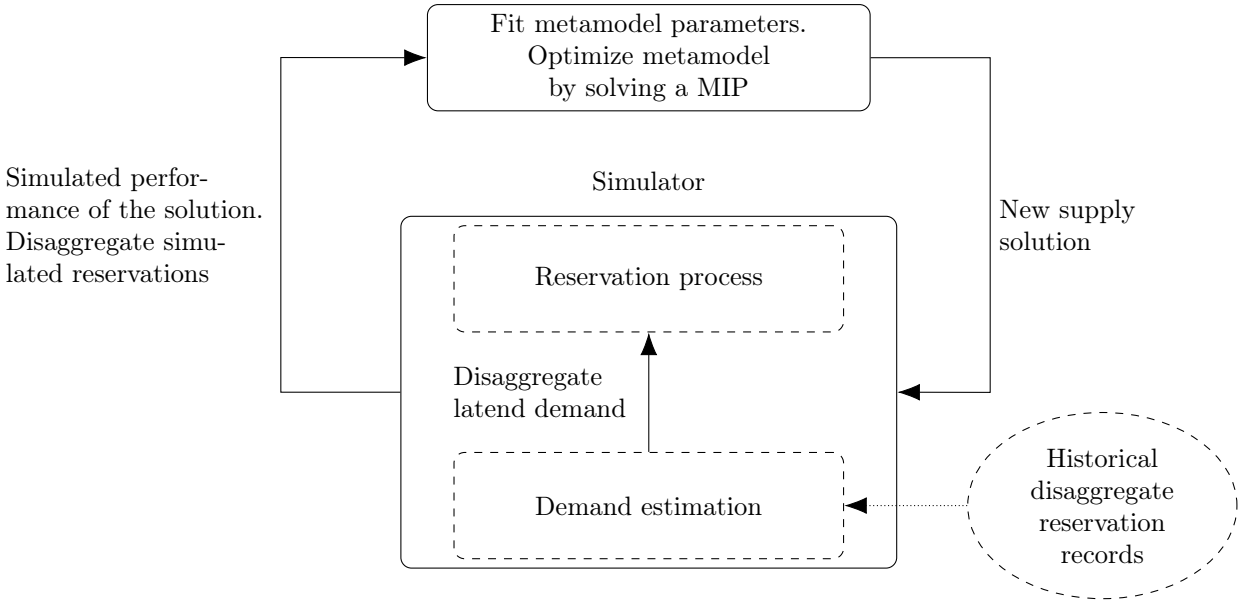


Figure 2-1: Data-driven metamodel SO framework

2.2.3 Car-sharing network design metamodel formulation

To formulate the analytical problem-specific component of the metamodel, g_A, which approximates the profit of a given network design strategy, we introduce the following additional notation.

- d^i_{tl}: number of customers that desire a reservation at station i with start time t and duration l;
- r_{tl}: revenue from a reservation with start time t and duration l;

- p^{ij}: discount to the revenue if a reservation is desired for station i but is fulfilled at (i.e., is made at) station j;
- z^i_{tl}: number of customers that make a reservation at station i with start time t and duration l;
- z^{ij}_{tl}: number of customers that desire to make a reservation at station i with start time t and duration l but make an adjusted reservation at station j with start time t and duration l;
- $t_{\max}$: number of one-hour reservation start time intervals during the planning period (e.g., for an n-day planning period, $t_{\max} = n \times 24$);
- $l_{\max}$: maximum reservation duration;
- $\mathcal{I}_i$: set of stations "near" station i, including station i;
- $\mathcal{L}$: set of reservation durations (in hours), $\mathcal{L} = \{1, 2, \ldots, l_{\max}\}$;
- $\mathcal{T}$: set of reservation start time interval indices, $\mathcal{T} = \{1, 2, \ldots, t_{\max}\}$;
- $\mathcal{T}_1(t, l)$: set of reservation start times for reservations with duration l that are ongoing at time t (i.e., they start prior to t and have not finished at time t).

The vector $\boldsymbol{z}$ defined in Section 2.2.2 consists of all variables $\{z^i_{tl}\}$ and $\{z^{ij}_{tl}\}$. The function g_A is formulated as:

$$g_A(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{q}_2) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} \sum_{t \in \mathcal{T}} \sum_{l \in \mathcal{L}} p^{ij} r_{tl} z^{ij}_{tl} - \sum_{i \in \mathcal{I}} c_i x_i. \tag{2.9}$$

This function is defined as the difference between the total revenue and the total cost. Note that in the total revenue expression, we give a discount (p^{ij}) for reservations that are adjusted (i.e., the initial desired reservation was not feasible because a car was not available). This allows us to account for the impact on revenue of demand spillback (i.e., demand censoring). Note that demand spillback and loss are described in a more detailed and disaggregate manner in the simulator (see. Section 2.2.5).

The auxiliary variable z_{tl}^{ij} is related to the decision vector $\boldsymbol{x}$ through the analytical network model, which is denoted by h in Equation (2.7) and is defined as follows.

$$\sum_{j \in \mathcal{I}_i} z_{tl}^{ji} = z_{tl}^{i} \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{2.10}$$

$$\sum_{j \in \mathcal{I}_i} z_{tl}^{ij} \leq d_{tl}^{i} \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{2.11}$$

$$\sum_{l \in \mathcal{L}} z_{tl}^{i} + \sum_{l \in \mathcal{L}} \sum_{t' \in \mathcal{T}_1(t,l)} z_{t'l}^{i} \leq x_i \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{2.12}$$

$$z_{tl}^{i} \in \mathbb{R}_+ \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{2.13}$$

$$z_{tl}^{ij} \in \mathbb{R}_+ \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}_i, \forall t \in \mathcal{T}, \forall l \in \mathcal{L}, \tag{2.14}$$

where $\mathcal{T}_1(t,l) = \{t' \in \mathcal{T} : t' + 1 \leq t \leq t' + l - 1\}$. Equation (2.10) states that z_{tl}^{i}, the number of reservations at station i with start time t and duration l, is the sum of all desired reservations at station j (with start time t and duration l) that were shifted to station i. Note that $i \in \mathcal{I}_i$, hence this summation includes the reservations that were desired and also made at station i (with start time t and duration l). Equation (2.11) is a demand constraint. The right-hand side is the total demand for station i with start time t and duration l. The left hand side considers the set of reservations with a preference for station i start time t and duration l. This summation includes reservations where: (i) the preference was available and was made, (ii) the preference was not available and the reservation was adjusted and made at a neighboring station j (with the same start time t and the same duration l). For a given fleet assignment, the difference between the right-hand side and the left-hand side represents the lost demand for reservations at station i with start time t and duration l. The left-side of the Constraint (2.12) consists of two terms. The first term represents the total number of reservations at station i that start at time t. The second term represents the total number of reservations at station i that have started prior to time t and are still ongoing. Hence, Constraint (2.12) ensures that at station i and time t, the number of reserved cars (left-side of the inequality) is bounded above by the number of cars assigned to station i. Constraints (2.13) and (2.14) assume non-negative

real values for the auxiliary variables (z^i_{tl} and z^{ij}_{tl}). The use of real-valued auxiliary variables, rather than integer variables, contributes to the computational efficiency of this analytical approximation. In this model, the exogenous parameters are d^i_{tl}, r_{tl}, c_i, p^{ij}, $t_{\max}$ and $l_{\max}$. Together they form the exogenous parameter vector $\boldsymbol{q}_2$. The endogenous variables are z^i_{tl}, z^{ji}_{tl} and x_i. The exogenous parameters r_{tl}, c_i, p^{ij} and $l_{\max}$ are directly estimated from the data and in consultation with Zipcar staff. Note that we have $0 \leq p^{ij} \leq 1$ and $p^{ii} = 1$ for all $i \in \mathcal{I}$ and $j \in \mathcal{I}_i$. A discussion on the simplifications of this analytical model compared to the simulator is given in Section 2.2.5.

The demand parameters (d^i_{tl}) are estimated by sampling from the historical reservation data to estimate latent demand. For a description of the demand sampling step, see Section 2.2.5. Since this data sampling is stochastic, the case studies of Section 2.3 consider experiments based on different realizations of the sampling (i.e., they consider different latent demand estimates). This serves to evaluate the impact of varying demand on the performance of the proposed method. Moreover, the SP, which is used as one of the benchmark algorithms in Section 2.3.4, considers a set of latent demand realizations. This data sampling process yields a set of disaggregate desired reservations, which are then aggregated to estimate the aggregate parameters d^i_{tl}. The simulator, unlike the analytical MIPs (metamodel MIP or SP), uses the disaggregate demand without aggregating it.

For any station $i \in \mathcal{I}$, if we assume the maximum number of stations in the neighborhood of i, $\mathcal{I}_i$, is smaller than a constant W, i.e., $|\mathcal{I}_i| \leq W$, then the number of auxiliary variables of the metamodel is in the order of $O\left(W|\mathcal{I}||\mathcal{T}||\mathcal{L}|\right)$, and the number of constraints is in the order of $O\left(|\mathcal{I}||\mathcal{T}||\mathcal{L}|\right)$. Hence, by bounding the duration of the planning period and the maximum duration of a reservation, the number of variables and the number of constraints increase linearly with the number of stations. This contributes to the scalability of the model. In summary, the metamodel optimization problem is a mixed-integer linear model, which can be solved in a computationally efficient way with a variety of standard solvers. For the case studies of Section 2.3.2, the MIP (2.6)-(2.8) is solved on average in 2.4 seconds for the the Boston South End

network and in 35.1 seconds for the large-scale Boston network of Section 2.3.3.

2.2.4 Discrete SO algorithm: MetaAHA

The proposed metamodel is embedded within a general-purpose discrete SO algorithm. We have chosen the Adaptive Hyperbox Algorithm (AHA) of Xu et al. (2013). As discussed in Section 2.1, AHA has been used to solve high-dimensional problems with a decision vector of dimension 100. AHA is a locally convergence random search algorithm. We use the term *current iterate*, denoted $\boldsymbol{x}_k$, to refer to the point considered to have best performance at iteration k. The name AHA stems from the sampling, at every iteration, from a region which is the intersection of the feasible region and the *hyperbox* (for more details, see Appendix B). The latter is centered at the current iterate with a size that is updated, at every iteration, based on the performance of the current iterate and of its neighbors. Let $\mathcal{H}_k$ denote the hyperbox at iteration k. The proposed algorithm, denoted MetaAHA, is an extension of AHA. Algorithm 1 presents MetaAHA.

Each iteration k of the algorithm consists of 4 main steps. Step 1 identifies the set of points to simulate. These can be new points that have not been simulated before or points that have already been simulated and for which we will run additional simulation replications. Step 2 simulates these points. Step 3 checks whether termination criteria are satisfied. Step 4 uses the set of all simulation observations collected so far and updates the fit of the metamodel. Additional algorithmic details and a flowchart summary of MetaAHA are given in Appendix B.

Algorithm AHA is obtained from MetaAHA by omitting Steps 1b, 1c , 3b and 4; and setting r (of Step 1a) to n (while for MetaAHA $r = n - 2$). Steps 1b and 1c solve mixed-integer programs. These steps yield solutions to MIPs. Hence, they exploit problem-specific analytical structural information provided by the analytical network design model. This information enables the algorithm to: (i) identify points with good performance within few, or even no, simulation runs because the analytical network design model can be solved without available simulation observations, and (ii) become less sensitive to the quality of the initial sample. This sensitivity to the

Algorithm 1 MetaAHA

Initialization:

– Initialize parameters: iteration index $k = 1$, hyperbox $\mathcal{H}_1 = \{\boldsymbol{x} : 0 \leq x_i \leq N^i, \forall i \in \mathcal{I}\}$, set n (the number of solutions to simulate per iteration). Set the number of randomly sampled solution in each iteration $r = n - 2$. For the metamodel parameter vector $\boldsymbol{\beta_1}$, set $\beta_{1,0} = 1$ and $\beta_{1,i} = 0$ $(\forall i \geq 1)$.

Step 1: identify the set of n points to sample

– Step 1a: obtain r points in $\mathcal{F} \cap \mathcal{H}_k$ based on the asymptotically uniform sampling mechanism of AHA.

– Step 1b: obtain 1 point, denoted $\boldsymbol{x}_k^{\text{meta}}$, as the solution to the metamodel optimization Problem (2.6)-(2.8).

– Step 1c: obtain 1 point, denoted $\boldsymbol{x}_k^{\text{meta-hyper}}$, as the solution to the metamodel optimization Problem (2.6)-(2.8) with the additional constraint that the point belongs to $\mathcal{H}_k$.

Step 2: simulation

– Following the procedure of AHA: simulate the points identified in Step 1; simulate $\boldsymbol{x}_{k-1}$ (for $k > 1$); select the point with best performance $\boldsymbol{x}_k$ (i.e., update the current iterate); update the hyperbox.

Step 3: check for algorithm termination

– Step 3a: test if $\boldsymbol{x}_k$ is a local optimum following the procedure of AHA. If so, stop.

– Step 3b: if the total number of iterations exceeds the maximum number of iterations (i.e., if the computational budget is depleted), stop.

Step 4: metamodel update

– Step 4a: for any simulated point $\boldsymbol{x}$ that has not been evaluated by the analytical network model, evaluate it (i.e., for a given $\boldsymbol{x}$, maximize $g_A(\boldsymbol{x}, \boldsymbol{z})$ of Equation (2.9) over $\boldsymbol{z}$ subject to Constraints (2.10)-(2.14)).

– Step 4b: use all simulation observations collected so far to fit the metamodel parameter $\boldsymbol{\beta}_k$ (i.e., solve the least squares Problem (A.1) defined in Appendix A).

Step 5: update iteration counter

– Set $k = k + 1$, proceed to Step 1.

quality of the initial sample has been identified and discussed in past AHA work (Xu et al., 2013).

While both Steps 1b and 1c exploit this problem-specific analytical information, Step 1c does so within the hyperbox, leading to the identification of local points with good performance, while Step 1b does so in the entire feasible region, leading to the identification of global points with good performance. In Step 2, we determine the number of replications to simulate for each point (this is done based on the approach of AHA, which is also described in Appendix B), we simulate the points and then update both the hyperbox and the current iterate. In Step 3b, if the computational budget is depleted, then the algorithm is terminated without convergence. This serves to reflect the most common way in which these algorithms are used in practice.

Note that MetaAHA does not change the main building blocks of the basic algorithm AHA. It merely complements it by adding a problem-specific sampling strategy which is based on the use of the metamodel. Hence, AHA's asymptotic local optimality guarantee is preserved. MetaAHA illustrates how a variety of general-purpose discrete SO algorithms can be complemented with such problem-specific sampling strategies to improve their robustness to the quality of the initial points as well as their short-term (i.e., small sample) performance. For practitioners, who typically use these algorithms under tight computational budgets, this has the potential to improve the performance of these general-purpose algorithms.

2.2.5 Two-way car-sharing simulator

We summarize here the main ideas underlying the simulator. For more details on the specification of the simulator as well as on its validation, see Fields et al. (2017). The simulator takes as input disaggregate historical reservation data, estimated daily demand per station (i.e., total daily number of reservations desired per station), a fleet assignment strategy, and yields as output a set of realized reservations (reservations actually made) with the corresponding network-wide profit.

More specifically, the simulation process consists of two main parts, as summarized in Figure 2-1. The first step, referred to as the demand sampling step, samples

from the data such as to (approximately) obtain a set of desired reservation requests (i.e., reservations that users would ideally desire to make). These reservations can be thought of as realizations of latent demand. Hence, we distinguish between realized demand (an empirical distribution of which is given by the dataset) and latent demand. The second step, referred to as the reservation simulation step, considers a given latent demand (i.e., a given set of desired reservations) and simulates the reservation process as follows. It ranks, and then sequentially processes, the desired reservations by increasing creation time. For a given reservation, if a car is available (at the desired station and during the desired time interval), then the reservation is made. Otherwise, with a given probability the client will either not make a reservation (this is referred to as lost demand) or it will consider an "adjacent" reservation, which is either at a nearby station or at a nearby start time (this is referred to as demand spillback; it accounts for demand censoring). The probability depends on the distance between the initially desired reservation and the considered adjacent reservation. Once a given reservation is made, other users cannot use the same car at any time during this reservation period. This procedure mimics the first-come-first-reserve process.

The most important input to the simulator is the set of historical disaggregate reservation data. In this work, we use Zipcar data. For each reservation observation in the dataset, the following attributes are used: station (this is both the pick-up and the drop-off location), start time, duration and reservation creation time (i.e., the timestamp of when the reservation was made). Additionally, based on information available online we have estimated reservation revenues. The time resolution of the simulator is based on that of the data which is 30 minutes. This means that reservation durations and reservation start times are defined in 30 minute increments.

A main feature of this simulator is that this reservation process simulation is based on a handful of parameters, which are estimated from the data. Additionally, there are few modeling assumptions, which were made in consultation with Zipcar staff. They include the probability of considering an adjacent reservation and the formulation of a distance metric between reservations. Each of the two steps of the simulation process

described above (i.e., demand sampling and reservation simulation) are stochastic. In other words, the generation of a set of desired reservation requests is stochastic and the mapping of a desired reservation to a realized (or even a lost) reservation is also stochastic.

We now present the main simplifications of the analytical network model compared to the simulator. These simplifications contribute to the formulation of an efficient analytical metamodel. First, the analytical model does not enforce the first-come-first-reserve rule of the simulator. In other words, for a given set of reservation requests, they will not be processed by increasing order of reservation creation time. Instead, the set of reservations that leads to highest (metamodel) profit will be realized regardless of their respective creation times. Second, the analytical model allows for reservations to be adjusted in space (i.e., change of station) but not in time (i.e., the start time of a desired reservation cannot change). Third, the adjustment process is simplified. For a given reservation, the simulator checks whether it is available, and if not with a certain probability it considers to either not make any reservation (leading to lost demand) or to attempt a nearby (in space and time) reservation (leading to demand spillback). The simulator iterates on these steps (i.e., a given client may attempt to make several reservations before deciding on a final reservation or before deciding not to make a reservation). In the analytical model, there is no sequential reservation process. Instead, demand spillback is approximated through the discounted revenue parameter, p^{ij} of Eq. (2.9). Fourth, the simulator considers a time resolution of 30 minutes (i.e., reservation start times and durations are defined in 30 minute increments), while the analytical model considers a time resolution of 1 hour.

2.3 Case studies

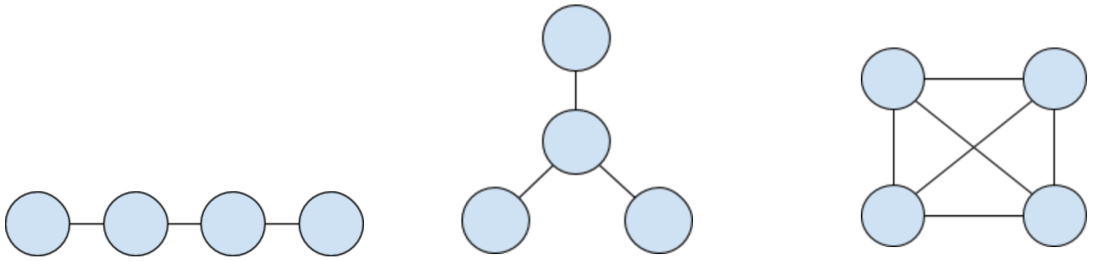
In this section, we apply MetaAHA to optimize the design of two-way car-sharing systems. Section 2.3.1 considers a low-dimensional problem with synthetic toy networks. Sections 2.3.2-2.3.4 consider high-dimensional problems for Zipcar's Boston market.

We study its two-way services for two Boston areas: (i) an area of downtown Boston known as South End (Section 2.3.2) and (ii) a larger network that includes 23 zipcodes of the Boston metropolitan area (they include Allston, Arlington, Boston, Brighton, Brookline, Cambridge, Charlestown, Chelsea, Medford and Somerville) (Section 2.3.3 and 2.3.4). All experiments are conducted on a machine with 125GB RAM with an Intel Xeon E5-2630 v3 processor.

2.3.1 Synthetic toy networks

The goal of these low-dimensional synthetic experiments is to evaluate the quality of the analytical approximation (g_A of Equation (2.9), which is provided by the analytical network model) of the simulation-based objective function (g of Equation (2.2)). We consider 3 networks with topologies that are simple and are representative of subnetwork topologies of Zipcar's Boston network. The 3 networks are displayed in Figure 2-2. Each circle represents a car-sharing station. Each network has four stations. Recall from Section 2.2.3 that, in the analytical model, when the desired reservation of a user is not available, he/she may consider a substitute chosen from a set of neighbors that are defined as spatially nearby locations (this set was denoted $\mathcal{I}_i$ in Section 2.2.3). In other words, these are stations where the demand can spillover. In Figure 2-2, for a given station i, its set of neighbors or substitute stations, $\mathcal{I}_i$, is the set of stations that are connected with an edge to station i. Hence, the three network topologies of Figures 2-2a, 2-2b and 2-2c consider, respectively, a loosely connected network of stations, where no stations share any neighbors; a centralized network, where all stations have the center station as a neighbor; and a fully connected network, where all stations are neighbors with all other stations. Each station has a capacity of 6 vehicles (i.e., N^i of Equation (2.4) equals 6), the fleet size is unlimited (i.e., X of Equation (2.3) takes any value such that $X \geq 24$). Hence, the feasible region is $\{x \in [0,6]^4 \cap \mathbb{Z}^4\}$, which contains 2401 feasible solutions. The data used for simulation is the Zipcar reservation data related to such a subnetwork. The planning period is an 8-day period in July 2014 (July 10 to July 17).

For each network, we generate a group of 10 demand scenarios. A demand scenario

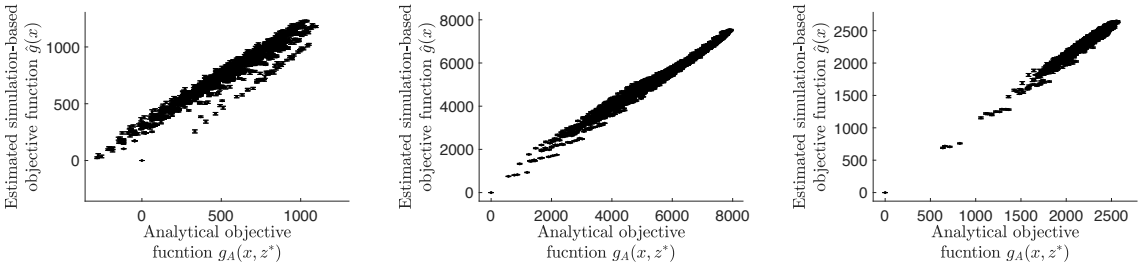


(a) Toy network 1 (b) Toy network 2 (c) Toy network 3

Figure 2-2: Toy network topologies

consists of the desired reservations generated through the demand sampling step described in Section 2.2.5. The use of various demand scenarios serves to account for demand stochasticity. For a given point, $\boldsymbol{x}$, one simulation replication (i.e., one simulation-based realization of its performance) is defined as the average simulated performance over the 10 demand scenarios. For a given point, $\boldsymbol{x}$, the final estimate of its simulation-based performance, $\hat{g}(\boldsymbol{x})$, is obtained as the average over 50 simulation replications. For the analytical model, we generate a different demand scenario to estimate its exogenous parameters (d^i_{tl} of Equation (2.11)). For a given point $\boldsymbol{x} \in \mathcal{F}$, the analytical objective function, $g_A(\boldsymbol{x}, \boldsymbol{z}^*)$, is obtained by maximizing Equation (2.9) over $\boldsymbol{z}$ subject to Constraints (2.10)-(2.14).

Each plot of Figure 2-3 considers one network and displays the analytical objective function, $g_A(\boldsymbol{x}, \boldsymbol{z}^*)$, along the x-axis and the estimated simulation-based objective function, $\hat{g}(\boldsymbol{x})$, with a corresponding 95% interval along the y-axis. The confidence intervals are barely visible. Each plot displays the 2401 feasible solutions.



(a) Toy network 1 (b) Toy network 2 (c) Toy network 3

Figure 2-3: Comparison of the analytical objective function value with the estimated simulation-based objective function value for toy networks

For all three plots, there is a positive linear correlation between the analytical approximations, $g_A(\boldsymbol{x}, \boldsymbol{z}^*)$, and the simulation-based estimates, $\hat{g}(\boldsymbol{x})$. This indicates that for all three representative network topologies the analytical network model provides a good approximation of the simulation-based objective function.

2.3.2 Boston South End network

We now consider the South End neighborhood in downtown Boston. A map of the area is displayed in Figure 2-4. The 23 stations over which we optimize are displayed with red circles. The planning period is July 10-17, 2014. During this period the average fleet size is 101 cars (i.e., $X = 101$). Based on consultation with Zipcar, we set the station capacity, N^i, to 16. We compare the performance of MetaAHA and AHA. This comparison serves to evaluate the added value of complementing AHA with information from the analytical problem-specific network model. The maximum number of algorithm iterations, K, is set to 40. At every iteration, the number of points to be simulated is set to 10 (i.e., $n = 10$).

To account for the stochasticity of demand, we proceed as in Section 2.3.1. We consider a group of 10 demand scenarios. For a given point, $\boldsymbol{x}$, one simulation replication (i.e., one simulation-based realization of its performance) is defined as the average simulated performance over the 10 demand scenarios. Figure 2-5 contains four plots. Each plot considers a different group of 10 demand scenarios. As in Section 2.3.1 for each group of demand scenarios, one additional demand scenario is used to estimate the exogenous parameters of the analytical network model.

Each plot displays the iteration index along the x-axis and the performance estimate of the current iterate (i.e., simulation-based estimate of the objective function of the best point) along the y-axis. The range of the y-axis differs across the plots. Each plot illustrates, for a given demand scenario group, the difference in performance of the two methods. Each plot displays 6 lines: 3 solid (resp. dashed) lines that represent 3 MetaAHA (resp. AHA) runs. For all plots, we observe the following main trends. First, MetaAHA identifies points with good performance from the first iteration, while the points initially sampled by AHA do not have good performance.

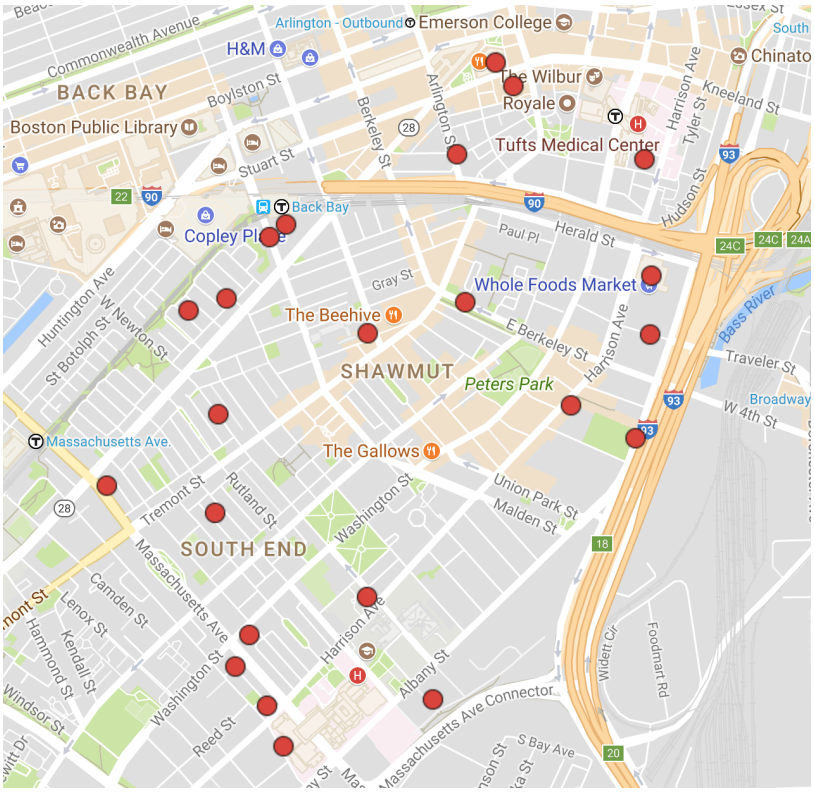
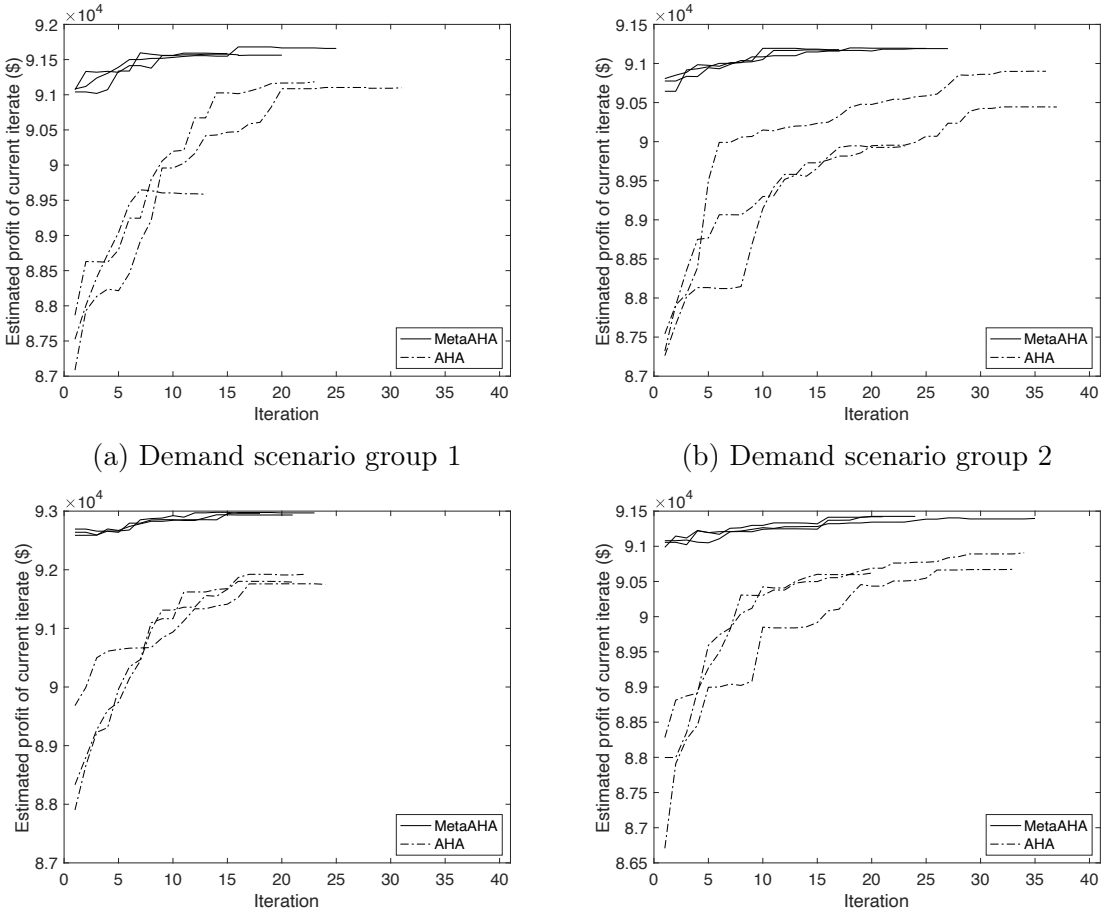


Figure 2-4: Zipcar stations in Boston South End neighborhood (map data: Google Maps (2017b))

Actually, for all six runs of MetaAHA, the best point identified in the first iteration corresponds to the solution of the analytical network design problem (i.e., maximize g_A of Equation (2.9) over both $\boldsymbol{x}$ and $\boldsymbol{z}$ subject to Constraints (2.3)-(2.5) and (2.10)-(2.14)). This shows the added value of the analytical structural information provided by g_A. Note that the initial points sampled by AHA are obtained from an asymptotically uniform sampling distribution for integral points from compact polyhedrons as defined in Hong and Nelson (2006). This general-purpose sampling method allows AHA to ensure asymptotic convergence properties, yet since it lacks problem-specific information, it is not designed to provide good quality initial solutions. Second, as the iterations advance, AHA identifies points with improved performance. This is consistent with the experiments and observations in Xu et al. (2013), which show that AHA is an efficient algorithm for a broad class of discrete SO problems. Nonetheless, it is outperformed throughout by MetaAHA. Third, MetaAHA shows a slight improvement across iterations, yet it is not as significant as that of AHA. Fourth, the



(a) Demand scenario group 1

(b) Demand scenario group 2

(c) Demand scenario group 3

(d) Demand scenario group 4

Figure 2-5: MetaAHA vs. AHA: objective function estimate of the current iterate across iterations

performance of the final solution derived by MetaAHA (i.e., the current iterate at the final iteration) is similar across the 3 MetaAHA runs, while final solutions have higher variability in performance for the 3 AHA runs. This indicates that MetaAHA is less sensitive to the stochasticity of the simulator. This may be attributed to the structural analytical information provided by the problem-specific network design model (g_A).

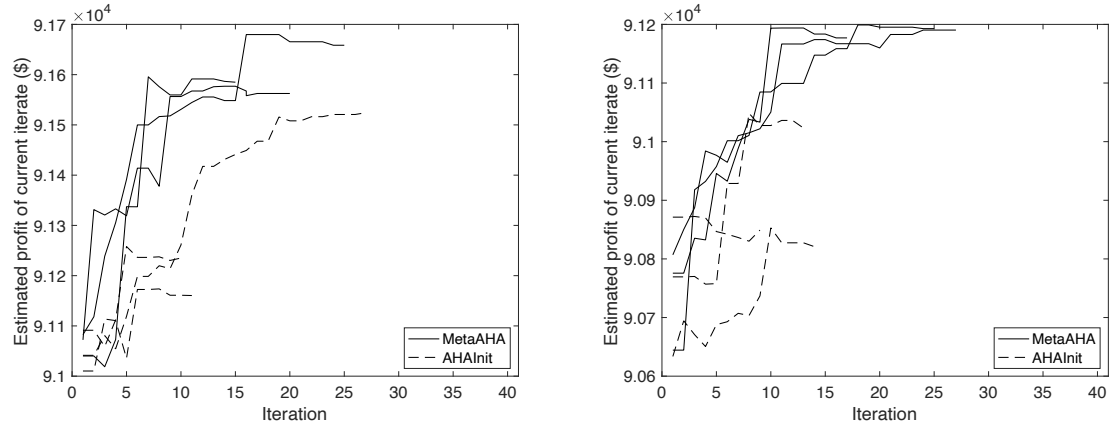
Note that in Figure 2-5 all lines terminate prior to iteration 40. This occurs if a current iterate is considered to be a local optimum (Step 3a of Algorithm 1). To limit the premature convergence of AHA, Xu et al. (2013) have combined it with the multi-start ISC framework (Xu et al., 2010). Also, most lines are not monotonically non-decreasing. This can occur when running additional simulation replications of

the current iterate leads to a lower objective function estimate (which can itself lead to a change of the current iterate).

These results indicate the ability of the metamodel approach to: (i) improve the robustness of the algorithm to the quality of the initial points, (ii) identify good solutions within very few iterations, and (iii) lead to low variability across the performance of the derived final solutions. These are all trends that have been observed in our past metamodel work for continuous SO transportation problems (Zhang et al., 2017, Chong and Osorio, 2017, Chen et al., 2019a, Osorio, 2019).

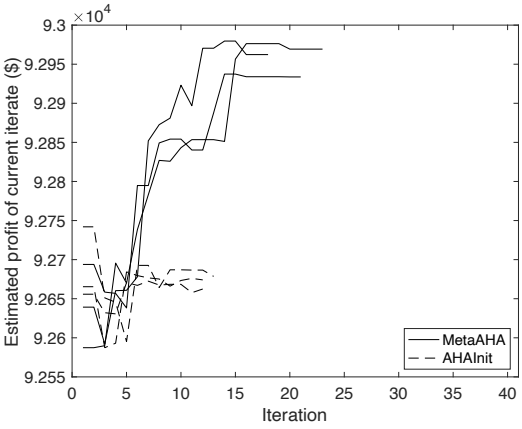
The results of Figure 2-5 indicate that a suitable approach would be to include in the initial sample of AHA the solution proposed by the analytical network design problem (i.e., the solution that maximizes g_A of Equation (2.9) over both $\boldsymbol{x}$ and $\boldsymbol{z}$ subject to Constraints (2.3)-(2.5) and (2.10)-(2.14)), and then to use the traditional AHA algorithm for all other iterations. Let AHAInit denote this approach. We now carry out a comparison of MetaAHA with AHAInit. This comparison serves to evaluate the added value of using analytical network model information across the iterations of AHA, rather than limiting the use of this analytical model to the first iteration. We use the same experimental design as for Figure 2-5. Figure 2-6 display four plots. Each plot considers a given group of 10 demand scenarios for the simulator and one demand scenario for the analytical model. The solid (respectively, dashed) lines represent MetaAHA (resp. AHAInit). The range of the y-axis differs across the plots.

The following trends are common to the four plots. First, MetaAHA outperforms AHAInit across all iterations. This reveals the added value of the metamodel m_k which combines the analytical network design information g_A with the simulation information. In other words, using the analytical network design model g_A to initialize a general-purpose algorithm contributes to its efficiency, yet there is even further added value of using the analytical information across iterations. Second, AHAInit tends to converge more quickly to a local optimum. Often, this local optimum has performance that is similar to that of the point obtained by solving the analytical network design problem (i.e., the point obtained by maximizing g_A subject to Constraints (2.3)-(2.5)

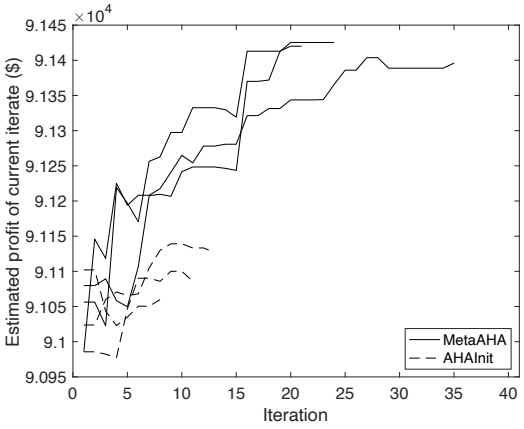


(a) Demand scenario group 1

(b) Demand scenario group 2



(c) Demand scenario group 3



(d) Demand scenario group 4

Figure 2-6: MetaAHA vs. AHAInit: objective function estimate of the current iterate across iterations

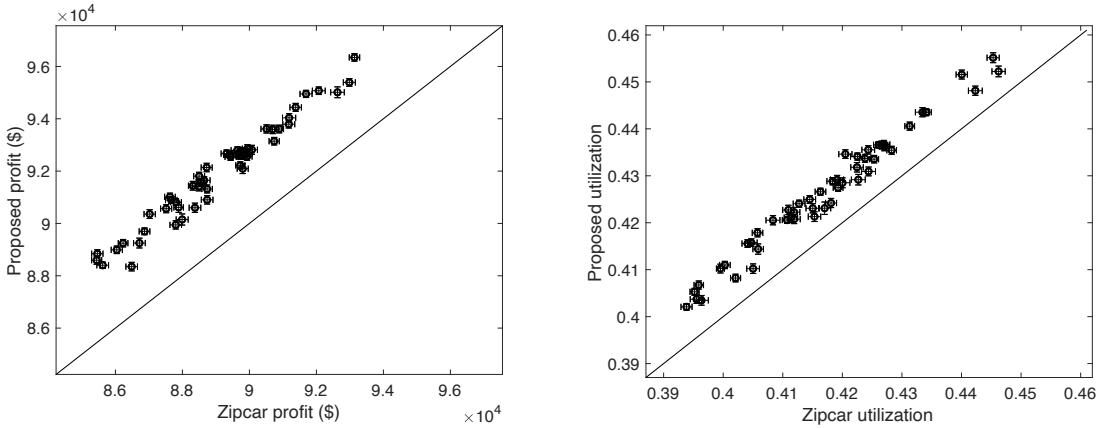
and (2.10)-(2.14)).

For the 12 MetaAHA runs of Figure 2-6 (i.e., 3 runs for each of the 4 plots), there are a total of 87 instances where the current iterate is updated. Recall that for MetaAHA a current iterate can be of 3 types: (i) it can be a solution to the metamodel optimization problem solved in the entire feasible region (i.e., Step 1b of Algorithm 1, which yields points denoted $\boldsymbol{x}^{\text{meta}}$), (ii) it can be a solution to the metamodel optimization problem solved in the intersection of the entire feasible region and the hyperbox (i.e., Step 1c of Algorithm 1, which yields points denoted $\boldsymbol{x}^{\text{meta-hyper}}$), or (iii) it can be obtained from random sampling (i.e., Step 1a of Algorithm 1, which yields points denoted $\boldsymbol{x}^{\text{sampled}}$). Note that a point can be both of type $\boldsymbol{x}^{\text{meta}}$ and of type $\boldsymbol{x}^{\text{meta-hyper}}$. This occurs when the solution to the metamodel optimization

problem in the entire feasible region is located in the hyperbox. Of the 87 different current iterates of the 12 MetaAHA runs in Figure 2-6, more than two thirds (i.e., 71.3% or 62 points) are of type $\boldsymbol{x}^{\text{meta}}$ or $\boldsymbol{x}^{\text{meta-hyper}}$, while less than one third (28.7% or 25 points) are of type $\boldsymbol{x}^{\text{sampled}}$. In other words, two thirds of the current iterates are obtained by using the structural information of the analytical network model. For the 12 final best solutions returned by the MetaAHA runs, 9 of them are identified by solving the metamodel and 3 of them by random sampling. Moreover for the 12 runs of MetaAHA, we simulated 2364 points. Only 18.1% of the simulated points are obtained by solving the metamodel (429 points), while the remaining 81.9% are obtained by random sampling. Hence, even though the points derived by metamodel evaluations represent only 18.1% of the total set of sampled points, they lead to 75% of the final solutions and 71.3% of the current iterates. This highlights the added value of the structural information provided by the analytical MIP. Among the 62 current iterates obtained by using structural analytical information, 21 are of type $\boldsymbol{x}^{\text{meta}}$ and 47 are of type $\boldsymbol{x}^{\text{meta-hyper}}$ (note that 6 points are both of type $\boldsymbol{x}^{\text{meta}}$ and $\boldsymbol{x}^{\text{meta-hyper}}$). This shows that both the global (i.e., in the entire feasible region) and the local (i.e., in the hyperbox) information of the analytical network model help to identify points with improved performance. Recall that the metamodel is fitted after every iteration, hence the metamodel optimization problems solved across iterations differ and hence their solutions may differ. It is through this fitting process that the metamodel combines information from the simulator with information from the analytical network model. The high number of distinct current iterates identified by the metamodels illustrates the added value, across iterations, of combining the analytical information with the simulated information.

Figure 2-7 compares the performance of the best fleet assignment identified by MetaAHA (the proposed strategy) with that used by Zipcar during the planning period of interest. The final proposed (or "best") MetaAHA solution is defined as follows. We consider a set of 50 new demand scenarios. For all 12 solutions derived by MetaAHA (i.e., 3 algorithmic runs for each of the 4 plots of Figure 2-6), we estimate the average (over the 50 demand scenarios, each scenario is simulated with

50 replications) performance. The proposed solution is that with the best (i.e., largest) average performance.



(a) Profit (b) Utilization

Figure 2-7: Comparison of the Zipcar fleet assignment with the proposed assignment for the Boston South End network

Figure 2-7 displays two plots. The left plot compares the profit estimates of the two assignments. The right plot compares them according to vehicle utilization. Both of these metrics are important for Zipcar. For each plot, the x-axis considers the Zipcar assignment and the y-axis considers the MetaAHA proposed assignment. Each plot displays 50 points, which correspond to 50 demand scenarios. For each demand scenario, we estimate the performance based on 50 simulation replications. The performance estimate of each point is displayed along with a, barely visible, 95% confidence interval along each direction. Both the left and the right plots indicate that for all 50 demand scenarios the proposed plan yields improved performance, and this across all 50 demand scenarios. Compared to Zipcar's fleet assignment, the proposed solution yields an average improvement of profit of 3.2% and of vehicle utilization of 2.2%. Recall that these estimates are obtained via simulation. Hence, they do not state that the proposed method outperforms the Zipcar method when deployed in the field.

2.3.3 Boston area network - comparison versus AHAInit

In this section, we consider a larger area of the Boston metropolitan area. This serves to evaluate the performance of MetaAHA for a high-dimensional problem. We consider a network of 315 stations distributed throughout 23 zipcodes that span over Allston, Arlington, Boston, Brighton, Brookline, Cambridge, Charlestown, Chelsea, Medford and Somerville. The map of Figure 2-8 displays the stations as red circles. We consider the same planning period as before. The station capacity, N^i, is set to 16, based on consultation with Zipcar. Historical data indicates that, during this planning period, there are an average of 894 cars assigned to these stations, i.e., $X = 894$. We proceed as before and consider a group of 10 demand scenarios. One additional demand scenario is used to estimate the exogenous parameters of the analytical model. We set the maximum number of iterations to 40 (i.e., $K = 40$) and the number of points to simulate per iteration to 70 (i.e., $n = 70$).

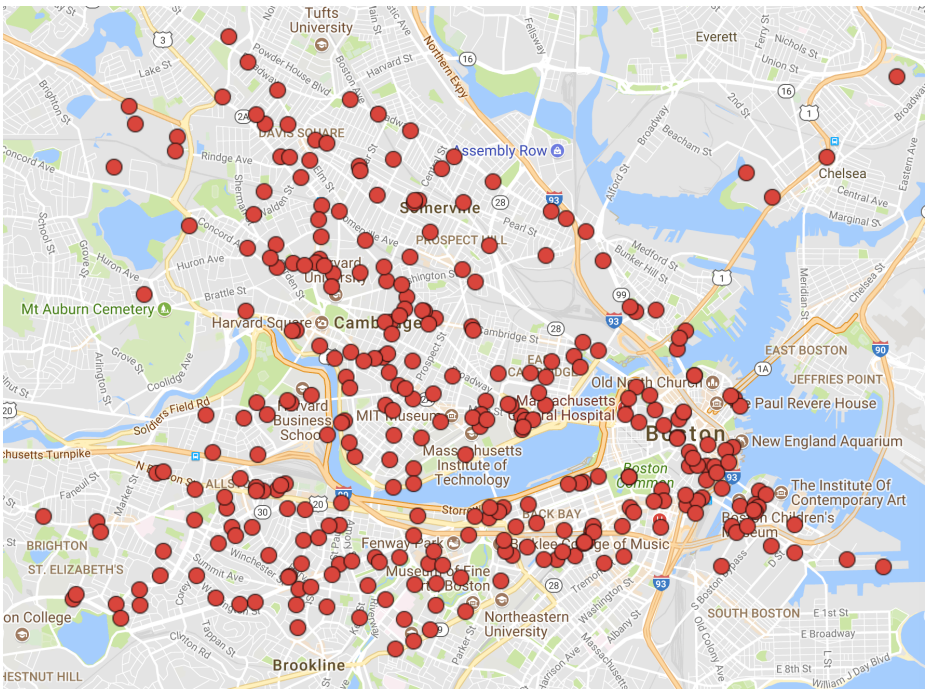


Figure 2-8: 315 Zipcar stations in Boston area (map data: Google Maps (2017a))

Figure 2-9 displays the results of 8 MetaAHA runs (solid lines) and 8 AHAInit runs (dashed lines). Only 3 of the 16 runs deplete the computational budget (i.e.,

they stop at iteration 40). They correspond to 3 MetaAHA runs. More specifically, the 8 MetaAHA runs stop at iterations 13, 14, 24, 33, 33, 40, 40 and 40. Those of AHAInit stop at iterations 14, 15, 15, 19, 20, 24, 33 and 38. All 8 runs of AHAInit yield final solutions with similar objective estimates. Seven out of the 8 MetaAHA final solutions are better than all 8 AHAInit final solutions.

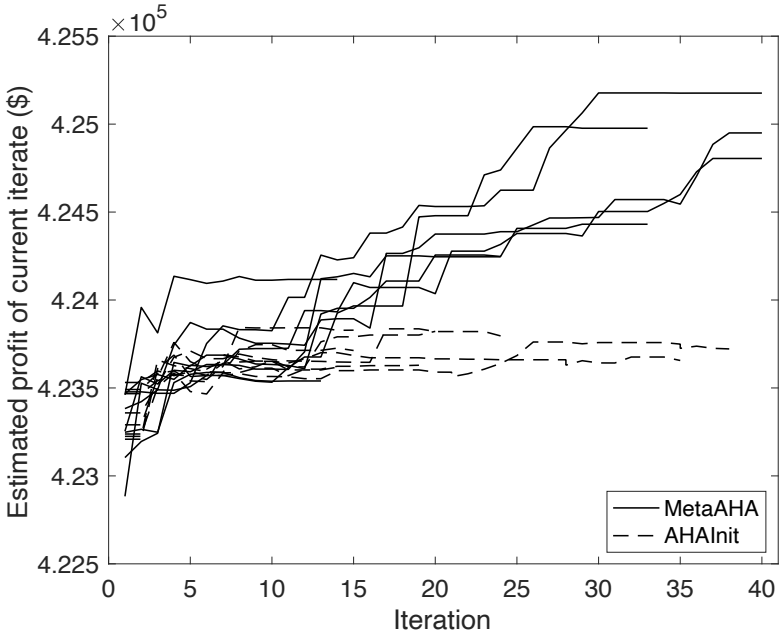
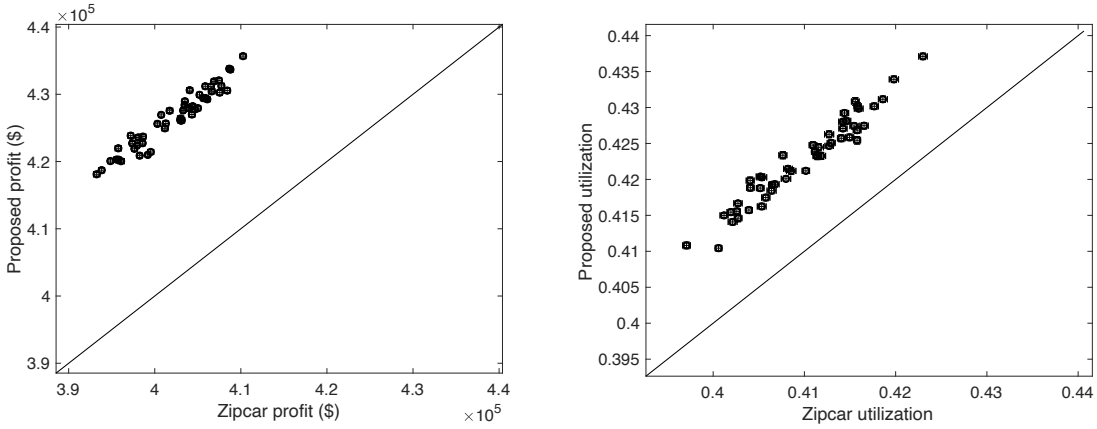


Figure 2-9: MetaAHA vs. AHA: objective function estimate of the current iterate across iterations

Figure 2-10 compares the performance of the best solution identified by MetaAHA with the fleet assignment strategy used by Zipcar. To evaluate the performance of a given fleet assignment strategy (that proposed by MetaAHA or that of Zipcar), we proceed as before. We generate 50 demand scenarios. For each of the 8 final solutions derived by MetaAHA and for each demand scenario, we run 50 simulation replications to estimate the average profit per solution. The solution with the highest average simulated profit is selected as the proposed solution. Figure 2-10 displays two plots: the left plot considers profit and the right plot considers vehicle utilization. For each plot, the x-axis considers the Zipcar assignment and the y-axis considers the proposed assignment. Each plot displays 50 points which correspond to the 50 demand scenarios. Each point estimate is displayed along with a 95% confidence interval along both directions. The confidence intervals, which are barely visible, are computed

based on 50 replications. For both plots, the 50 points, which represent 50 different demand scenarios, are above the diagonal. Compared to Zipcar's fleet assignment, the proposed solution yields an average improvement of profit of 6% and of vehicle utilization of 3.1%. Moreover, for all 50 demand scenarios, the proposed strategy improves both the profit and the fleet utilization. Again, note that this comparison is based on simulated performance, which may not reflect field performance.



(a) Profit (b) Utilization

Figure 2-10: Comparison of the Zipcar fleet assignment with the proposed assignment for the Boston area network

2.3.4 Boston area network - comparison versus stochastic programming

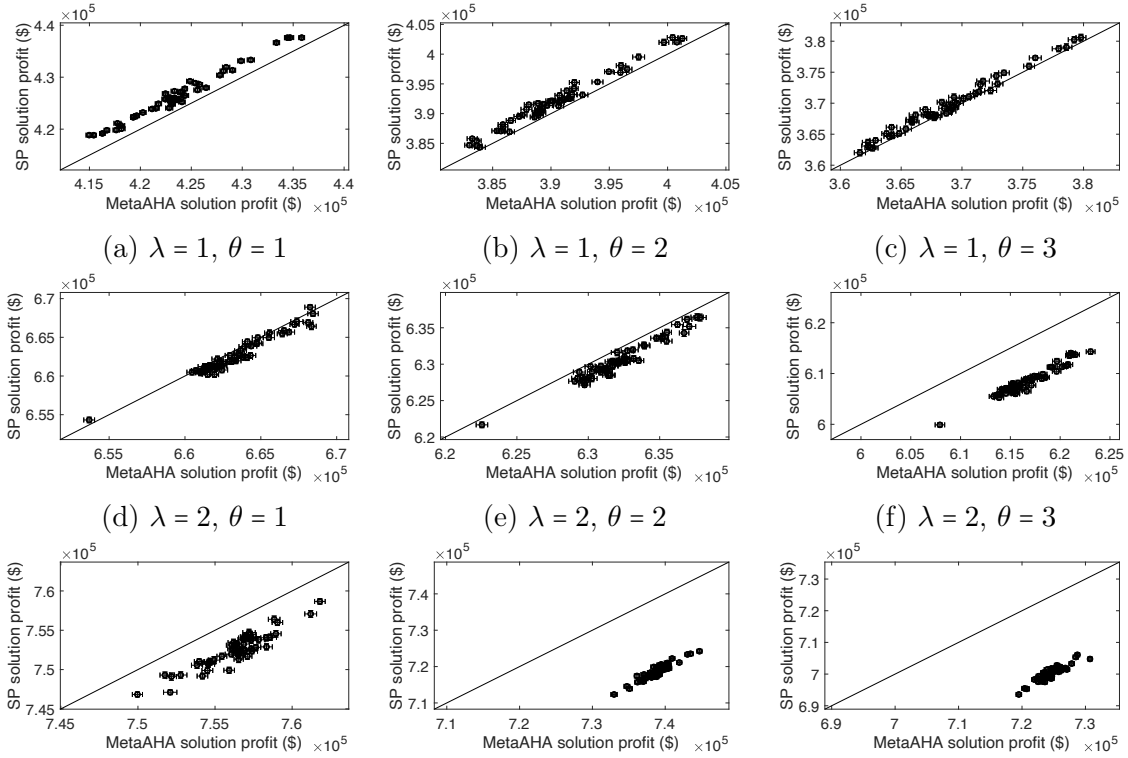
As mentioned in Section 2.1, the most common approach to study the fleet assignment problem is the use of analytical optimization methods, such as mathematical programs. In this section, we benchmark the performance of the proposed approach to stochastic programming (SP), which accounts for demand uncertainty. We consider a two-stage SP, where the second stage accounts for demand scenario realizations. The SP formulation is given in Appendix C. We use the same Boston area network as that of Section 2.3.3. We consider a set of 9 experiments with varying levels of demand and of cost. Demand is scaled by a factor: $\lambda \in \{1, 2, 3\}$, cost (i.e., term c_i of Eq.(2.2)) is scaled by a factor: $\theta \in \{1, 2, 3\}$. For each experiment (i.e., a given

value of (λ, θ)), we proceed as follows. We consider one demand scenario to estimate the exogenous parameters of MetaAHA, and 3 additional demand scenarios for SP and for the simulator. Hence, the SP model and the simulator used as part of MetaAHA have the same demand input. When evaluating a solution via simulation (within MetaAHA), the simulated estimate of the objective function is obtained as the average over the 3 demand scenarios. We solve the SP model to obtain one SP solution. We run MetaAHA 3 times (in order to account for the stochasticity of both the simulator and the algorithm) to obtain 3 MetaAHA solutions. We use the same MetaAHA algorithmic parameters as in Section 2.3.3 (i.e., $K = 40$ and $n = 70$). For each final solution (SP or MetaAHA), we simulate its performance considering 50 demand scenarios and 50 simulation replications per demand scenario (for a total of 2500 simulations per solution). We compare the performance of the SP solution to that of the best MetaAHA solution, which is defined as that with the best simulated profit over the 2500 simulations.

Figure 2-11 displays 9 plots, one for each experiment. The demand scaling factor λ (resp. cost scaling factor θ) is constant across columns (resp. rows) and increases across rows (resp. columns). Each plot displays 50 points, which correspond to each of the 50 demand scenarios. The y-axis (resp. x-axis) displays the estimated, via simulation, mean profit over 50 replications using the SP (resp. best MetaAHA) solution. Each point has a 95% confidence interval along both coordinate directions. These intervals are barely visible. Each plot also displays the diagonal line defined by $y = x$.

For $\lambda = 1$ (i.e., top row of plots: Figures 2-11a, 2-11b, 2-11c), all 50 points are above the diagonal line. In other words, the SP solution outperforms that of MetaAHA. More specifically, the SP solution improves, on average, the profit by 0.65%, 0.43% and 0.21%, for $\theta = 1, 2$ and 3, respectively. This trend is reversed for the other 2 rows of plots. For $\lambda = 2$ (i.e., second row of plots), MetaAHA outperforms SP by an average of 0.1%, 0.26% and 1.36%, for $\theta = 1, 2$ and 3, respectively. For $\lambda = 3$ (bottom row of plots), MetaAHA outperforms SP, on average, by 0.51%, 2.80% and 3.48%, for $\theta = 1, 2$ and 3, respectively. For demand levels λ of 2 or 3 (i.e., second or third

row of plots), as the cost level θ increases, so does the amount by which MetaAHA outperforms SP. For a given cost level (i.e., a given column of plots), as the demand level increases (i.e., from the top row to the bottom row), so does the amount by which MetaAHA outperforms SP.



(a) $\lambda = 1$, $\theta = 1$ (b) $\lambda = 1$, $\theta = 2$ (c) $\lambda = 1$, $\theta = 3$

(d) $\lambda = 2$, $\theta = 1$ (e) $\lambda = 2$, $\theta = 2$ (f) $\lambda = 2$, $\theta = 3$

(g) $\lambda = 3$, $\theta = 1$ (h) $\lambda = 3$, $\theta = 2$ (i) $\lambda = 3$, $\theta = 3$

Figure 2-11: Comparison of the average profit, considering 50 demand scenarios, of the SP solution and of the best MetaAHA solution

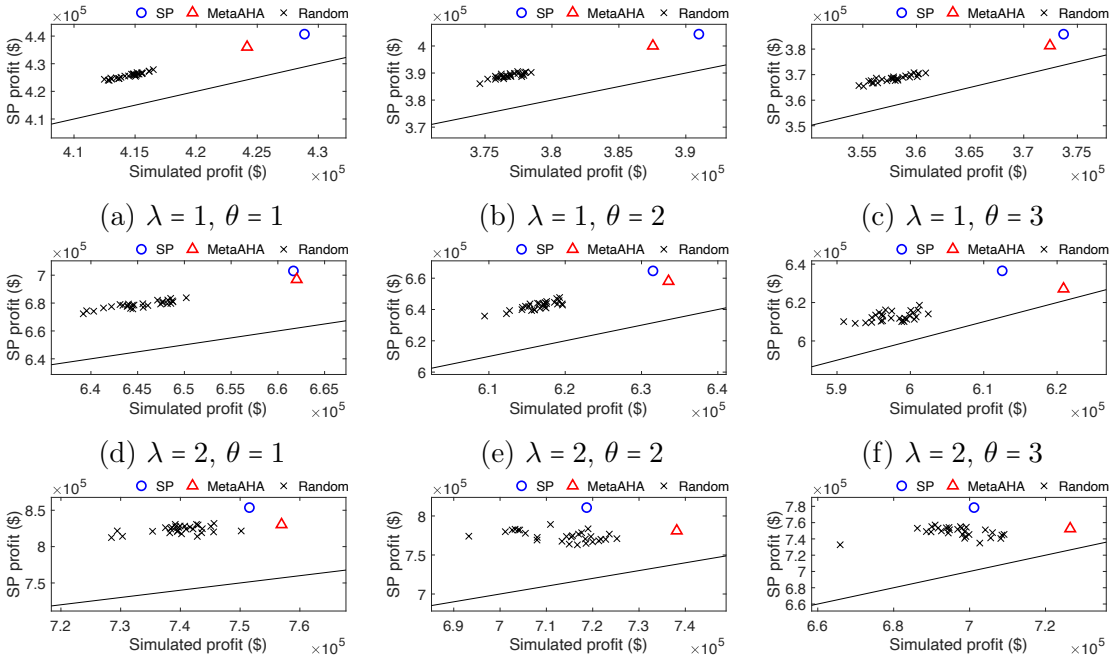
Figure 2-12 considers the same 9 levels of demand and cost. It evaluates the ability of SP to approximate the simulation-based objective function. Each point of Figure 2-12 considers a given feasible solution and displays along the x-axis the simulated estimate of its objective function value and along the y-axis its SP objective function value. Each plot considers a set of the following 30 feasible solutions: the SP optimal solution (displayed as a blue circle), the best MetaAHA solution (displayed as a red triangle), and a set of 28 randomly sampled solutions (black crosses) that are in the neighborhood of the line connecting the SP optimal solution and the best MetaAHA solution (the sampling process is detailed in Appendix D). The diagonal

line, $y = x$, is also displayed. For each solution, the SP objective function value and the simulation-based objective function estimate are based on the same 3 demand scenarios. These demand scenarios are the same as those used to derive the SP solution of the previous analysis. The simulation-based objective function estimate is obtained as the average across 50 replications of each of these 3 demand scenarios (i.e., each estimate involves $150 = 50 \times 3$ simulation evaluations). Figure 2-13 differs from Figure 2-12 in that the y-axis of each plot displays the metamodel objective function value. The value of the metamodel parameter $\boldsymbol{\beta}$ is that of the last iteration of the MetaAHA run which generated the best MetaAHA solution. Each of the 9 subplots of Figure 2-12 have the same axis limits as the corresponding subplot in Figure 2-13. Hence, the subplots are directly comparable across Figures 2-12 and 2-13.

For all plots of Figure 2-12, all points are above the diagonal line, i.e., SP tends to overestimate the simulated profit for all cost and demand levels. For $\lambda = 1$, the SP objective function exhibits a positive linear correlation with the simulated estimate. This also occurs for $(\lambda, \theta) \in \{(2,1),(2,2)\}$. Hence, the SP model correctly ranks the performance of the feasible solutions, and hence is an adequate tool for optimization. Nonetheless for high values of demand and of cost (i.e., all plots with $\lambda = 3$, as well as the plot $(\lambda, \theta) = (2,3)$), there is no longer a positive linear correlation.

For all plots of Figure 2-13, all points are close to the diagonal line. This indicates that the metamodel approximates well the simulation-based objective function. The positive linear correlation also highlights the ability of the metamodel to correctly rank the performance of the solutions, and hence its adequacy for optimization. Unlike Figure 2-12, this correlation trend holds even for high levels of demand and of supply.

These figures indicate the ability of the metamodel approach to approximate the simulation-based objective function even as the demand-supply interactions become more intricate to model (i.e., when both demand levels and cost levels are high). Recall that the main simplifications of the MIPs (both the metamodel MIP and the SP) compared to the simulator are the lack of the first-come-first-reserve principle, as well as the coarse description of demand spillback (for a description of these sim-

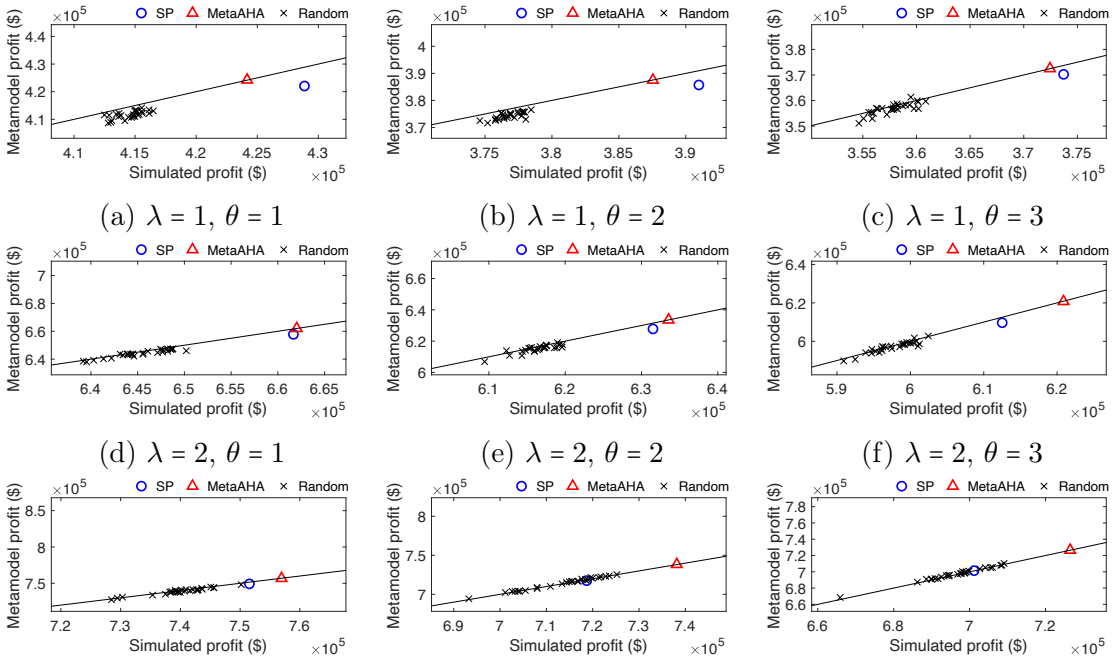


(a) $\lambda = 1, \theta = 1$ (b) $\lambda = 1, \theta = 2$ (c) $\lambda = 1, \theta = 3$

(d) $\lambda = 2, \theta = 1$ (e) $\lambda = 2, \theta = 2$ (f) $\lambda = 2, \theta = 3$

(g) $\lambda = 3, \theta = 1$ (h) $\lambda = 3, \theta = 2$ (i) $\lambda = 3, \theta = 3$

Figure 2-12: Comparison of the objective functions of the SP model and of the simulation model across various demand levels and cost levels.



(a) $\lambda = 1, \theta = 1$ (b) $\lambda = 1, \theta = 2$ (c) $\lambda = 1, \theta = 3$

(d) $\lambda = 2, \theta = 1$ (e) $\lambda = 2, \theta = 2$ (f) $\lambda = 2, \theta = 3$

(g) $\lambda = 3, \theta = 1$ (h) $\lambda = 3, \theta = 2$ (i) $\lambda = 3, \theta = 3$

Figure 2-13: Comparison of the objective functions of the metamodel and of the simulation model across various demand levels and cost levels.

plifications, see the last paragraph of Section 2.2.5). Hence, we expect the ability of the MIPs to approximate the simulator's objective function to deteriorate as the demand and cost levels increase. This is illustrated by comparing Figures 2-12 and 2-13. Moreover, the results of Figure 2-13 indicate that a simple linear parametric correction to the MIP (through the metamodel parameter $\boldsymbol{\beta}$) suffices to correct for these simplifications. In other words, we need not resort to the formulation of a more intricate analytical optimization problem (e.g., with nonlinear functions to describe spillback in more detail).

2.3.5 New York City case study

In this section, we apply our method to the Manhattan area of New York City. The goal of this study is to detect the impact of different total car supply levels and different demand levels on the performance metrics of a car-sharing company. We use the data from Zipcar. We focus our study during April, May and June of 2017.

The goal of the experiments is know the performance of a car-sharing service under different demand levels and supply levels. Given a specific level of demand and a specific level of supply, we solve the optimization problem (2.2) to (2.5) to find an optimized network design strategy. We evaluate this strategy using the simulator, and treat its performance as the performance of the car-sharing service under the given demand and supply level.

Experiment Setup

We design the following experiments for a single month. We can then apply the same experiments to each of the three months, with only data input changed.

To study the impact of different levels of demand and supply on the performance of a car-sharing service, we will vary value of the demand vector d and fleet size X. For a month, let d_0 be the demand vector we trained from data using the method of Fields et al. (2017) and X_0 be the actual average fleet size of the month. Note that the number of cars at a station may change during the month, so to obtain X_0, we

calculate the average number of cars at each station and sum them up.

For each month, we will use three different demand levels, and denote the set of these levels as $\mathfrak{D} = \{90\% \times d_0, d_0, 110\% \times d_0, \}$. We will also use three different supply levels, and deote the set of these levels as $\mathfrak{S} = \{80\% \times X_0, 90\% \times X_0, X_0\}$. Hence, there are 9 different combinations of demand and supply levels. For each such combination $(d, X) \in \mathfrak{D} \times \mathfrak{S}$ ($d \in \mathfrak{D}$ and $X \in \mathfrak{S}$), we find an optimized network design strategy, and use the simulator to evaluate 7 metrics: profit, total hours served, fleet utilization, total number of customers served, total number of customers lost, revenue obtained and revenue lost. Note that the optimization objective function is just the mean simulated profit.

For different months, we use the same c_i, N^i ($\forall i \in \mathcal{I}$), and q across all experiments. We assume all cars are homogeneous. In this study, there are 277 Zipcar stations, i.e., $I = 277$.

Input preparation

For each of $d \in \mathfrak{D}$, we generate and store the following intermediate data:

1. using the historical reservation records of the study month, generate $Q_1 = 21$ scenarios, 20 of which will be used as simulator input, and the rest one will be used to generate demand parameter for the analytical model;

2. using the historical reservation records of the study month, generate one group of $Q_2 = 50$ scenarios, which will be used to evaluate the solutions;

Experiments Procedure

Here we present the experiment procedure and some modifications of MetaAHA. For each combination of $(d, X) \in \mathfrak{D} \times \mathfrak{S}$, we do the following things.

1. Read in all previously simulated solutions that were obtained in any previous experiments with the same demand vector d. Denote the set with all those solutions as $\mathcal{F}_0$. Note some of the solutions may not be feasible current in for current supply X.

2. Randomly generate $N = 80$ feasible solutions, and add them into $\mathcal{F}_0$

3. For each solution $x \in \mathcal{F}_0$, simulate 10 replications and evaluate it analytically, store the simulated objective value and analytical objective value.

4. Run MetaAHA once. In the procedure of MetaAHA, when fitting the meta-model's objective function in each iteration, we use all solutions in $\mathcal{F}_0$ and those newly obtained in the current iteration. The fitting center is the current best feasible solution obtained in this MetaAHA run. For solutions newly obtained in the current iteration will be added to $\mathcal{F}_0$. If a solution is already in $\mathcal{F}_0$, just updated the newly obtained simulation results.

5. After MetaAHA finishes, for each solutions in $\mathcal{F}_0$. Store the solution vector (x), all simulated profit and analytical profit.

For the MetaAHA run, we set the maximum number of iterations to be 20, and in each iteration, we sample 48 solutions randomly from the current hyperbox. In the procedure of MetaAHA, if solution x is to be simulated at iteration k, we compute the number of simulation replications to run at this iteration as $\max\{N_k(x) - a(x), 0\}$, where $a(x)$ is the total number of replications that x has been simulated in this MetaAHA run and previous experiments with same demand vector d, and $N_k(x) = \min\{10, \lceil 10(\log_{20}(k))^{1.01} \rceil\}$.

Select the Best Solutions

For each combination (d, X), usually several thousands of solutions will be simulated after the MetaAHA stops. Even though we can sort these solutions by their mean simulated profit value, it should be noted that the solution with the highest ranking we find in this way may not be the actual best one among all feasible solutions that have been simulated. This is because (i) the small number of simulation replications we run for each solution (ii) the $Q_1 = 20$ demand scenarios for the simulator may not be representative enough to show the whole picture of demand distribution.

Another issue is that Zipcar divide the Manhattan island into 23 different non-overlapping zones. Each station belongs to a specific zone. The feasible solution for (d, X) we obtained are station-based, namely, each element of the decision vector is the number of car assigned to a station. We can aggregate it into a zonal solution - a vector with each element standing for the number of car in a zone. This is done by simply sum up the numbers of cars assigned to the stations in a zone. Hence, one zonal solution may correspond to several station-based solutions.

To select the best solution for a combination of (d, X), we order all feasible solutions (station-based) we obtained in the MetaAHA procedure by their mean simulated profit. Let $\mathcal{F}_{dX} = \varnothing$ be the set for candidate solutions. Beginning from the one with the largest mean simulated profit, we add each solution into $\mathcal{F}_{dX}$ unless (i) $\mathcal{F}_{dX}$ has 10 solutions, (ii) there are three different corresponding zonal solutions for those solutions already $\mathcal{F}_{dX}$ and (iii) there are five solution in $\mathcal{F}_{dX}$ whose the zonal solution is the same as the zonal solution of this solution. Hence, $\mathcal{F}_{dX}$ will have at most 10 solutions after this process. These solutions corresponds to no more than three zonal solutions. Each zonal solution correspond to fewer than 5 solutions.

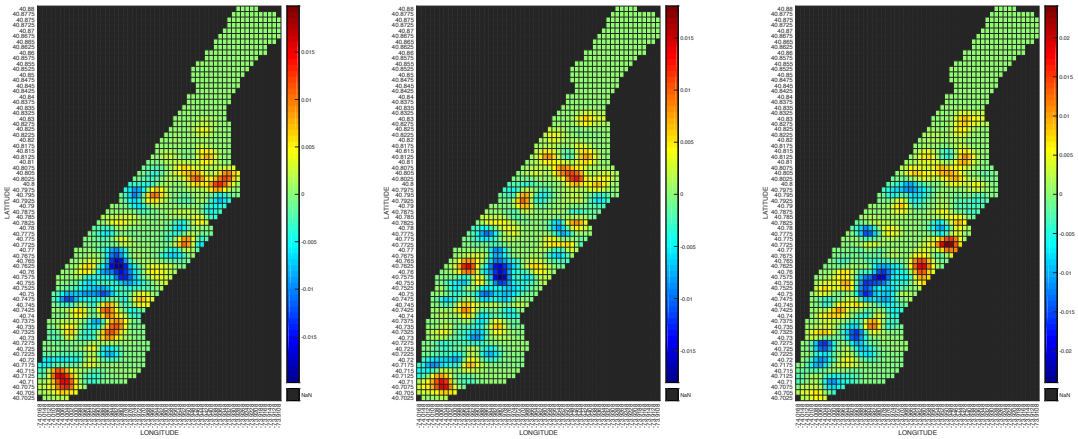
For the station-based solutions in $\mathcal{F}_{dX}$, we simulation each of them for 50 replications, using the $Q_2 = 50$ demand scenarios we generated before as the input for the simulator. For each solution, we plot its empirical cumulative distribution function of the simulated profit based on the 50 replications. Then we pick one using the curves, and treat it as the best solution for combination (d, X).

Experiment Results

We present some interesting results. First, we show the comparison between the optimized solution and actual network design strategy Zipcar used. In order to have a fair comparison, we only consider cases with trained demand and actual fleet size for each month (100% demand level and 100% supply level).

The heat maps of Manhattan in Figure 2-14, we show the different of the optimized strategy and actual strategy for April, May and June of 2017, respectively. The shape of a heat map approximate the Manhattan island. The x-axis is the longitude and the

y-axis is the latitude. The color of the heat map indicate the change of cars. Green means either there is no Zipcar stations, or the optimized solutions suggests do not change the number of cars. For places where the color is more red, the optimized solution suggests to add more cars to this places. For places where the color is more towards blue, we suggests to remove more cars. The numbers on the color bar on the right side of each heat map are used in the process of generating the heap maps. Their values do not make practical sense when we interpret the heat maps. The big blue area in the lower center part, which is areas around the Empire State Building, shows that during the three-month period, Zipcar seems put more cars than needed there. This is consistent with our observation that the fleet utilization there is low. The optimized solution suggests that removing cars from there can increase expected profit. The upper center part of the maps, which is the East Harlem area, shows red for the three-month period. This is consistent with our observation that the fleet utilization there is high. These comparisons shows that our method gives consistent suggestions over the time.

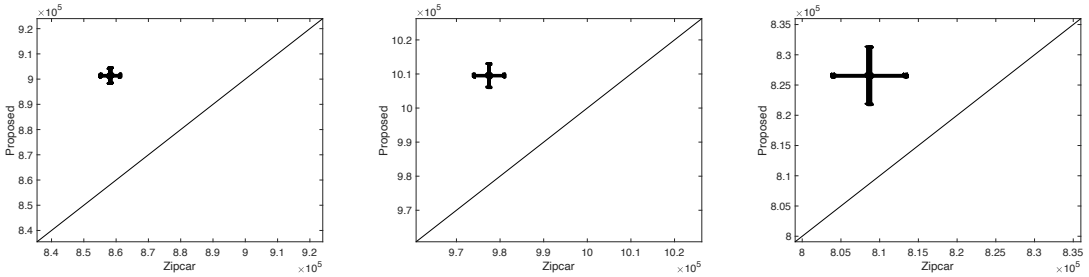


(a) 2017 April (b) 2017 May (c) 2017 June

Figure 2-14: Optimized Solution vs. Real Assignment (100% Supply 100% Demand)

In Figure 2-15, we compare the simulated profit of the proposed solution under the 100% demand level and 100% supply level and the Zipcar actually fleet assignment for each month. Each of the three plots correspond to a month. The x-axis is profit of Zipcar's actual assignment and the y-axis is the performance of the proposed solution.

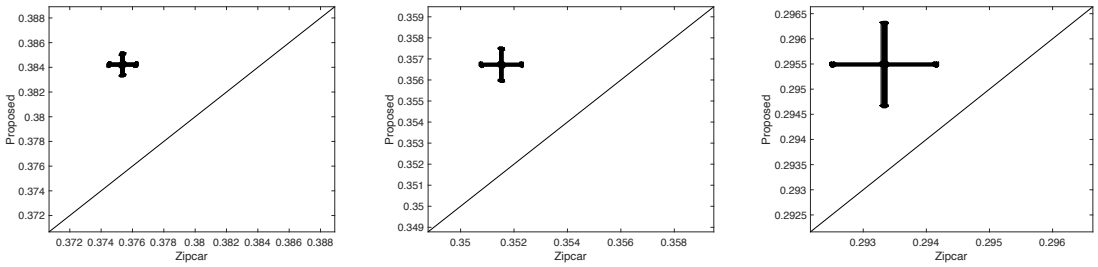
There are 50 points in each plot, and each corresponds to one demand scenario. For each solution, we simulate each demand scenario for 50 times. The vertical error bar on each point stands for a 95% confidence interval of the mean profit estimation of the corresponding demand scenario under the proposed solution. Similarly, the horizontal error bars is for the profit confidence intervals under Zipcar's actual fleet assignment. For each month, we can see that all 50 points is above the diagonal line. Also, all error bars do not intersect with the diagonal line. These results indicates that the simulated profit of the proposed solution is significantly higher than the simulated profit of actual fleet assignment. In Figure 2-16, we make a similar comparison for the fleet utilization of the two network design strategies. The results is also similar to the comparison of profit: for each month, all 50 points are all above the diagonal line and all error bars do not intersect with the diagonal line. Note only maximizing the profit, rather than utilization, is the objective of the network design problem. The comparison of utilization indicate that our solution can significantly increase the utilization of the fleet from the actual practice.



(a) 2017 April (b) 2017 May (c) 2017 June

Figure 2-15: Profit Comparison (100% Supply 100% Demand)

Another interesting finding shows that overall Zipcar may have a fleet size larger than actually needed. Here we focus on June 2016. In Figure 2-17, we plot the profit of the optimized solution under different demand levels and supply levels. Different colors stand for different level of demand. The x-axis stand for the level of supply. The y-axis stand for the profit. In Figure 2-18, we plot the revenue of the optimized solution under different levels of demand and supply. The plot format is same as Figure 2-17, except that the y-axis is the revenue. We can see that as the supply



(a) 2017 April (b) 2017 May (c) 2017 June

Figure 2-16: Utilization Comparison (100% Supply 100% Demand)

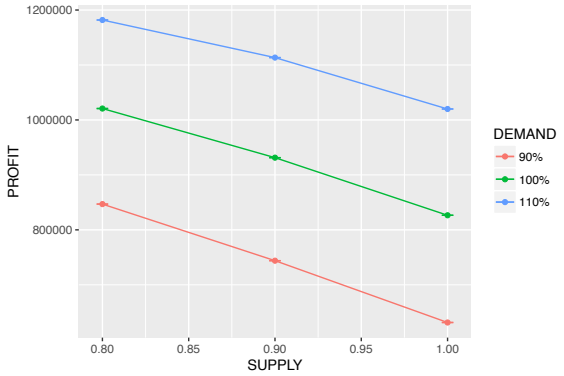


Figure 2-17: June 2016 Profit

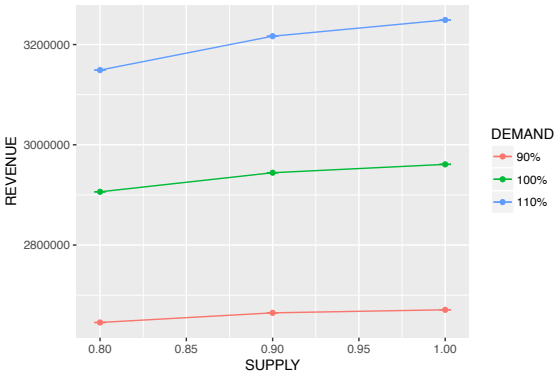


Figure 2-18: June 2016 Revenue

increase, for different levels of demand we tried in experiment, the revenue increases slightly, while the profits decrease. This means that the as we increase the supply level from 80% to 100%, the increase of revenue cannot cover the extra cost of adding more cars. Which means the 100% supply level is larger than needed for even 100% demand level. This finding results in the conclusion that Zipcar put too many cars in the Manhattan market for June 2017. The same phenomenon can be also observed in April and May of 2017.

In Figure 2-19, we plot the change of total reservation hours served; in Figure 2-20, we plot the change of number of customers served. We can see that both reservation hours and number of served customers increase only slighted as the supply increase, under all three demand cases. In Figure 2-21, we plot the change of fleet utilization as the supply level increases for all three demand levels. We can see that as the supply level increases, the fleet utilization drops quickly. This indicates that if we add more cars to the Manhattan fleet during June 2017, the fleet resources might be wasted.

These findings are consistent with our findings with revenue in Figure 2-18.

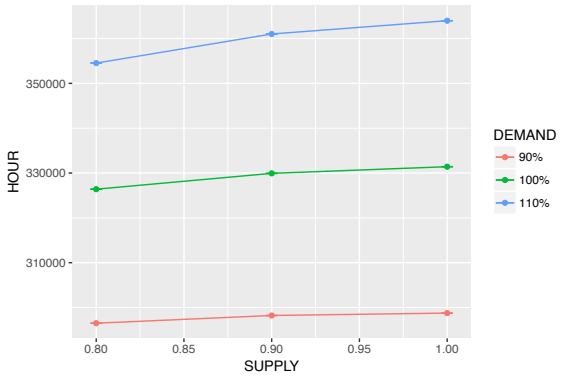


Figure 2-19: June 2016 Hour

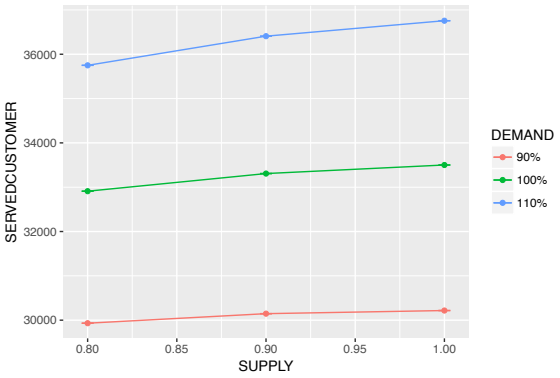


Figure 2-20: June 2016 Served Customer

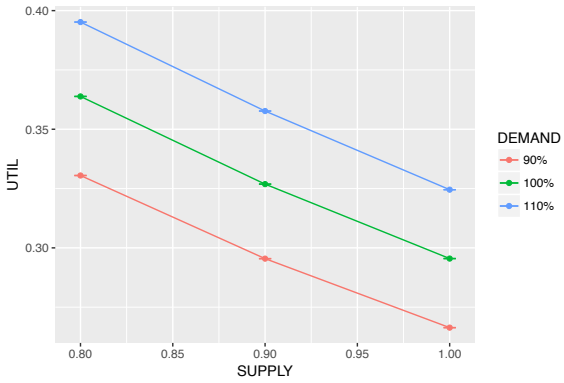


Figure 2-21: June 2016 Utilization

2.4 Summary

This chapter formulates a discrete SO algorithm for a family of large-scale car-sharing network design problems. The approach is a metamodel SO approach. A novel metamodel is formulated, which is based on a MIP formulation. The metamodel is embedded within a general-purpose locally convergent discrete SO algorithm.

The proposed algorithm is validated with synthetic toy network experiments. The metamodel approximations of profit are shown to have a positive linear correlation with the higher resolution simulation-based profit estimates. The algorithm is then applied to several Boston case studies using Zipcar car-sharing reservation data, including a high-dimensional problem. The method is first benchmarked versus two

types of algorithms that differ only in their use of the analytical MIP information: one benchmark algorithm (AHA) does not use any analytical network information (i.e., no MIP information), the second benchmark algorithm (AHAInit) uses the MIP information only to identify an initial solution but not throughout the optimization process. The experiments indicate that the analytical network model information is useful both at the first iteration of the algorithm and across iterations. The solutions derived by the proposed method are also benchmarked versus the Zipcar deployed solution. Via simulation, the proposed solutions outperform those deployed, both in terms of profit and vehicle utilization. This holds for all considered demand scenarios.

We also benchmark MetaAHA versus stochastic programming (SP). SP outperforms the proposed approach for low levels of demand and of cost. As demand and cost levels increase, so does the occurrence of demand spillback and the importance of accounting for the first-come-first-reserve principle. In these cases, the SP approach is outperformed by the metamodel approach.

The combination of the problem-specific analytical MIP with a general-purpose SO algorithm enables the discrete SO algorithm to: (i) address high-dimensional problems, (ii) become computationally efficient (i.e., it can identify good quality solutions within few simulation observations), (iii) become robust to the quality of the initial points and to the stochasticity of the simulator. More generally, the information provided by the MIP to the SO algorithm enables it to exploit problem-specific structural information. Hence, the simulator is no longer treated as a black box.

Chapter 3

A globally convergent discrete SO algorithm suitable for high-dimensional car-sharing service design problems

3.1 Introduction

In this chapter, we propose to extend a globally convergent discrete SO methods to enable its efficient use for high-dimensional car-sharing service design problems. The proposed approach allows us to utilize the detailed high-resolution car-sharing service data. We reviewed recent studies about car-sharing network design in Section 2.1.1.

Over the past years, there are many studies about globally convergent discrete SO algorithms. Proposed by Yan and Mukai (1992), stochastic ruler algorithm is one the the earliest globally convergent random search discrete algorithm. It creates a discrete-time Markov chain over the problem feasible region. In each iteration, this algorithm explore a small neighborhood around the current iteration. This algorithm only maintain simulation results of one solution, which allows it save memory spaces in implementation. But its practical performance is usually poor (Hong et al., 2015).

Andradóttir (1999) improved the stochastic ruler algorithm, and showed that using accumulated information can largely improve the performance of random search algorithms. Shi and Ólafsson (2000) introduced the Nested Partition algorithm (NP), which partition the feasible region into non-overlapping subregions. In each iteration, NP partition the current best subregion and sample and simulate solutions from each of the newly generated subregions. NP uses a bounding method that rank subregions by comparing the best solution in each of them. Andradóttir and Prudius (2009) extended NP and proposed Balanced Explorative and Exploitative Search with Estimation (BEESE), which is a framework combines a global search scheme based on NP and a local improve scheme based on hill-climbing algorithm. Norkin et al. (1998) proposed Stochastic Branch-and-Bound (SBB) algorithm. This method also relies on partition the feasible region into non-overlapping subregions. Unlike NP, which aggregate most existing subregions except the best one into a large subregion, the SBB algorithm keeps all subregions. In this way, the algorithm does not need to record how each subregion are generated, which saves machine memory when implementing. Based on SBB, Xu and Nelson (2013) proposed Empirical Stochastic Branch-and-Bound (ESBB), which combines the branching scheme of SBB and the bounding method of NP. This is the state-of-art random search discrete SO algorithm. However, most these algorithms have only been tested on low-dimensional problem with less than 30 decision variables. Shi and Ólafsson (2000) provided a case study of a travel salesman problem (TSP) with 101 cities (i.e., 101 decision variables), in which the objective value can be computed in a very fast way.

Another type of algorithms rely on an analytical model to obtain solutions. These methods mainly have two steps in one iteration: sample solutions based on the model, and update the model. Some algorithms use probabilistic models which assign a probability to be sampled to each feasible solution. Hu et al. (2008) proposed to use distributions belongs to the natural exponential family. Other commonly used probabilistic model includes Gaussian process (e.g., Sun et al. (2014)) and Gaussian Markov random fields (e.g. Salemi (2014) and Salemi et al. (2019)). Xie et al. (2016) proposed a globally convergent Bayesian optimization framework using Gaus-

sian model and Kriging model. Metamodel SO is another type of model-based SO algorithm. A metamodel is an analytical model which approximates the simulated objective value of the SO problem (Barton and Meckesheimer, 2006). Commonly used metamodels for SO include Kriging model (Kleijnen, 2009, Ankenman et al., 2010, Xu, 2012), radial-basis model Wild et al. (2008) and low-order polynomial model. See Barton and Meckesheimer (2006) for a summary. These general-purpose discrete SO algorithms do not embed any problem specific information. They are referred as functional metamodel Søndergaard (2003). They have been tested with problem with fewer than 20 decision variables.

There is a lack of literature about globally convergent discrete SO algorithms that can solve high-dimensional problem (i.e., with more than 10 decision variables according to Xu et al. (2013)) efficiently under tight computational budget. In practice, many problems require to obtain good solutions within given time. Most asymptotically globally convergent algorithm are not designing for high-dimensional problems and are not scalable due to the curse of dimensionality. Hence, there is a need to develop methods that can provide good solutions at the earlier stage of the algorithm run and maintain the globally convergent property so that the algorithm does not stop immaturely.

In this work, we propose an algorithm that combines metamodels with ESBB, a state-of-art globally convergent random search general-purpose discrete SO algorithm. We follow the framework of Osorio and Bierlaire (2013) and use metamodels with problem specific information. We study the problem of how to use analytical model that contains problem specific information to enhance the computational efficiency and scalability of a globally convergent discrete SO algorithm.

We use the metamodel for two purpose: (1) update and solve the metamodel in each iteration to obtain new solutions, and (2) use the metamodel to build a smaller sampling region and enhance the efficiency for solution sampling. We have two different metamodels, both in the form of MIP. We will study the performance of the proposed algorithm using the two metamodels of different accuracy. In this study, we have detailed data about the historical reservation record of a two-way car-sharing

service provider. We use an efficient simulator propose by Fields et al. (2017), which can sample directly from the data to approximate the demand information. By using this simulator, we achieve the goal of optimizing the car-sharing network design using disaggregated information from historical data.

The following part of this paper are organized as follows. In Section 3.2, we discuss our methodology, including the metamodel SO framework, two metamodels of different accuracy and the globally convergent algorithm we propose. In Section 3.3, we present case studies to show how well the metamodels can approximate the simulator, and the performance of the proposed algorithm. In Section 3.4, we summarize this chapter.

3.2 Methodology

In this chapter, we use discrete SO algorithm to address the problem of two-way car-sharing service design, the same problem we addressed in Chapter 2. We determine the way to assign certain number of cars to given stations in a way that the expected profit of the car-sharing system will be maximized. A station is a predetermined place where the company put a certain number of cars. We first formally define the problem as a SO problem. This formulation is the same as the one proposed in Section 2.2.1. We use the following notation.

- I: total number of stations
- $\mathcal{I}$: set of all stations, $\mathcal{I} = \{1, 2, ..., I\}$
- $\boldsymbol{x}$: decision vector, x_i $(i \in \mathcal{I})$ is the number of cars at station i
- $R(\boldsymbol{x}; \boldsymbol{q}_1)$: random variable representing the revenue
- $g(\boldsymbol{x}; \boldsymbol{q}_1)$: expected profit, i.e., the objective function of the SO problem
- c_i: cost of putting one car at station $i \in \mathcal{I}$
- $\boldsymbol{q}_1$: exogenous simulation parameters

- N^i: capacity of station i
- X: total fleet size, i.e., the number of cars to be assigned
- $\mathcal{F}$: problem feasible region

We formulate the problem as the follows

$$\max_{\boldsymbol{x}} \quad g(\boldsymbol{x}; \boldsymbol{q}_1) = E[R(\boldsymbol{x}; \boldsymbol{q}_1)] - \sum_{i \in \mathcal{I}} c_i x_i \tag{3.1}$$

subject to

$$\sum_{i \in \mathcal{I}} x_i \leq X \tag{3.2}$$

$$x_i \leq N^i \quad \forall i \in \mathcal{I} \tag{3.3}$$

$$x_i \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}. \tag{3.4}$$

The objective function (3.1) is the expect profit, i.e., the expected revenue $E[R(\boldsymbol{x}; \boldsymbol{q}_1)]$ minus the cost $\sum_{i \in \mathcal{I}} c_i x_i$, when the network design is represented by the decision vector $\boldsymbol{x}$. For any given $\boldsymbol{x}$, the revenue $R(\boldsymbol{x}; \boldsymbol{q}_1)$ cannot be computed directly, rather, we can only obtain one sample by running the simulator for one replication. The expected revenue for a given network design $\boldsymbol{x}$ can be estimated by calculating the average value over samples obtained by running multiple simulation replications. Constraint (3.2) requires that the total number of cars assigned be smaller than the fleet size. Constraint (3.3) forces the number of cars at each station be smaller than the station capacity. Constraint (3.4) means that all decision variables should be non-negative integers. Constraint (3.2)-(3.4) forms the problem feasible region $\mathcal{F}$.

We use the two-way car-sharing simulator of Fields et al. (2017). This simulator samples directly from historical reservation records to represent the attempts demand information, and then try to assign these attempts to the car-sharing fleet. For any given network design strategy, it will return the reservation attempts succeed, and compute the system revenue based on the succeeded attempts. One drawback of such data is that a lot of lost customers cannot be observed when the car supply is low. To

overcome this problem, Fields et al. (2018) discussed in detail about how to address the censored and truncated historical reservation, and provided a framework to infer the latent demand using the simulator of Fields et al. (2017) that samples directly from historical reservation record of two-way car-sharing service. We use the demand inferred by that framework. When supply is low, the order of the reservation making of different customers also matters a lot. In Section 2.1.1, we defined the term *first-come-first-serve* to describe the principle that the customer who make reservation first will get the car if there are others wanting to use a car at the same place and for overlapping reservation times.

In the following part of this section, we introduce the general metamodel SO framework in Section 3.2.1. Then introduce the two metamodel we will use in Section 3.2.3. Next, we introduce the low-dimensional search subregion construction method and its benchmark methods in Section 3.2.4

3.2.1 Using the MIP as a metamodel

As discussed in Section 3.1, a metamodel is an analytical model that approximates the simulator's objective function. Metamodel simulation-based optimization, i.e., metamodel SO, is a procedure that iteratively solve and update the metamodel. Osorio and Bierlaire (2013) proposed a framework that use a metamodel which combines a problem specific metamodel, or physical metamodel, with a lower-order polynomial model. Such combination was used to solve a continuous SO problems in urban transportation context. It increases the computational efficiency of general-purpose algorithms (Osorio and Nanduri, 2015, Chong and Osorio, 2017, Zhang et al., 2017, Chen et al., 2019a, Osorio, 2019). Zhou et al. (2019) (i.e., Chapter 2) extended the framework of Osorio and Bierlaire (2013) and proposed MetaAHA, an locally convergent algorithm, and applied to solve large-scale car-sharing network design problem. Such algorithm are efficient to find good solutions for high-dimensional discrete SO problems.

We follow the framework of Osorio and Bierlaire (2013). We use the following notation.

- k: the iteration index of the SO algorithm
- m_k: the metamodel at iteration k
- $\boldsymbol{\beta}_k$: vector of metamodel parameters at SO iteration k
- β_i: the ith element of $\boldsymbol{\beta}_k$
- $\boldsymbol{z}$: vector of endogenous variables
- $\boldsymbol{q}_2$: vector of exogenous parameters
- g_A: analytical expression approximating g, the simulated objective function
- h: constraints of the analytical model

Following the idea of Osorio and Bierlaire (2013), the metamodel of iteration k is as follows.

$$\max_{\boldsymbol{x},\boldsymbol{z}} \quad m_k(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\beta}_k,\boldsymbol{q}_2) = \beta_{k,0} g_A(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2) + \left(\beta_{k,1} + \sum_{i\in\mathcal{I}} \beta_{k,i+1} x_i\right) \tag{3.5}$$

$$h(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2) = 0 \tag{3.6}$$

$$\boldsymbol{x} \in \mathcal{F}. \tag{3.7}$$

The objective function (3.5) approximates the expected profit - the simulator's objective function g (i.e, Equation (3.1)). It consists two parts. The first part $\beta_{k,0} g_A(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2)$ is an analytical approximation $g_A(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2)$ of g multiplied by a scale parameter $\beta_{k,0}$. g_A is the physical metamodel which has problem specific information. It describes g in a simplified way. In our case, g_A is a MIP that approximate the simulated profit. The second part $\left(\beta_{k,1} + \sum_{i\in\mathcal{I}} \beta_{k,i+1} x_i\right)$ is a linear expression of all decision variables. It is the functional part of the metamodel. In each iteration of the algorithm we will present later, the value of the metamodel parameters (i.e., $\boldsymbol{\beta}_k$) will be updated by the fitting methods described in Appendix A. Hence, in each iteration, the metamodel will have different parameter values, and solving it may result in different solutions. Equation (3.6) represents the set of all constraints needed for the metamodel. Such a

constraint can be either equality or inequality. Since in optimization problem, we can always change an inequality constraint into equality constraints by adding auxiliary variables, we simple use one equality constraint to represent them. Equation (3.7) indicates that $\boldsymbol{x}$ must be feasible, i.e., subject to Equation (3.2)-(3.4). Details about the metamodel of our car-sharing network design problem will be presented in Section 3.2.3.

Compared to the original SO problem (3.1)-(3.4), the metamodel no long need the simulator to evaluate its objective function. It is now an analytical optimization problem. Also, we add more constraints to the original problems, i.e., Equation (3.6).

In Figure 3-1, we summarize the basic logic of metamodel framework. At the beginning of each iteration, we update the metamodel parameter $\boldsymbol{\beta}$ using previous simulated solutions. Detailed description about how to update the value for $\boldsymbol{\beta}$ are presented in Appendix A. We obtain new solutions by both solving the metamodels and randomly sampling. Then we feed the newly obtained solutions to the data-driven simulator, and evaluate their performance. For each solution, the simulator will return disaggregated simulated reservation, based on which we can estimated the expected revenue. With the revenue estimation, we enter the next iteration and update the metamodel again. In this work, we focus on the step of how to obtain better solutions more efficiently.

3.2.2 Using the MIP to identify low-dimensional subregions with good performance

Often, globally convergent discrete SO algorithms are not designed for high-dimensional problem. For example, an important step of ESBB is to partition the current best subregion into lower-dimensional non-overlapping subregions and then sample from the newly generated subregions. When the number of decision variables is large, the volume of the current best subregion will still be huge even after many iterations. Uniformly sampling within such a large space will be inefficient. In our car-sharing problem, there is a network effect: demand of one location may spillback to nearby

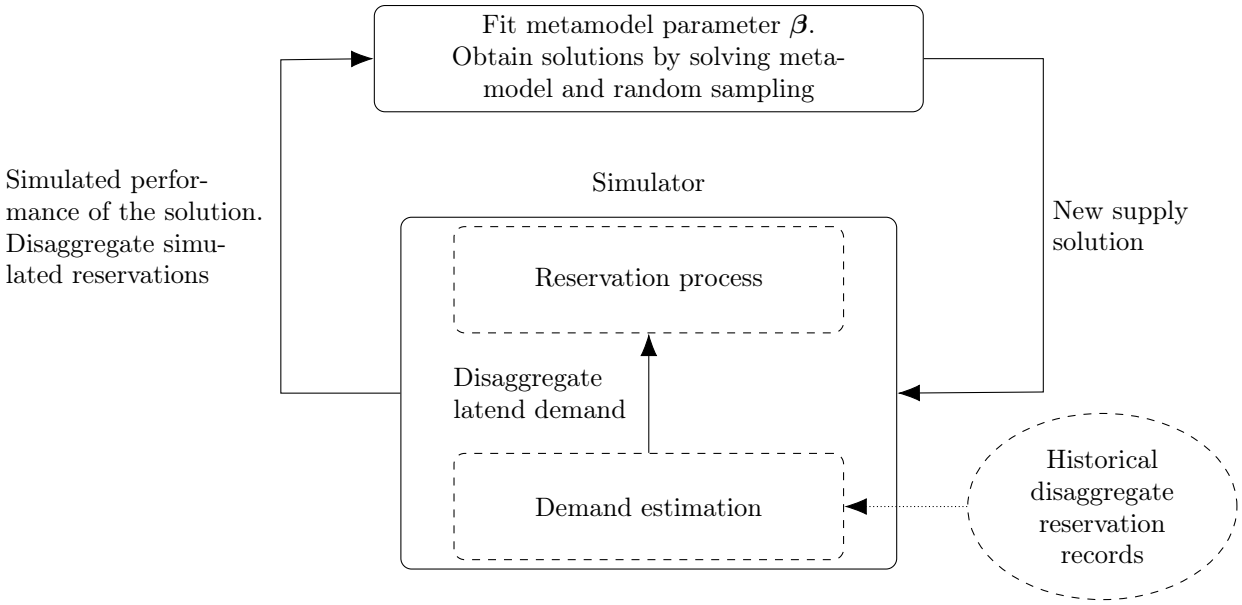


Figure 3-1: Data-driven metamodel SO framework

locations. We propose a method that utilize such information to define the subregions for sampling We use a metamodel to select the dimensions. Then we build a small hyperbox around the given solution, such that only values of the selected dimensions are allowed to be changed, while the rest dimensions are fixed.

To formulate the dimension-selection problem, we use the following additional notation.

- $\boldsymbol{x}^k$: the current iterate at the end of iteration k
- $\mathcal{R}^k$: the current best subregion, i.e., the subregion containing $\boldsymbol{x}^k$
- LB_i^k: the minimum value of all solutions in $\mathcal{R}^k$ along dimension $i \in \mathcal{I}$, i.e., $LB_i^k = \min_{\boldsymbol{x} \in \mathcal{R}^k} x_i$
- UB_i^k: the maximum value of all solutions in $\mathcal{R}^k$ along dimension $i \in \mathcal{I}$, i.e., $UB_i^k = \max_{\boldsymbol{x} \in \mathcal{R}^k} x_i$
- $\boldsymbol{u}$: a binary decision vector, $u_i = 1$ if dimension i ($i \in \mathcal{I}$) is selection, otherwise $u_i = 0$
- $\boldsymbol{u}^k$: the solved value of $\boldsymbol{u}$ in iteration k

- U: the maximum number of dimensions to be selected, a non-negative integer
- ε: a small positive real number

At the beginning of iteration k ($k > 1$) of MetaESBB, we have $\boldsymbol{x}^{k-1}$ as the current iterate, $\mathcal{R}^{k-1}$ as the current best subregion, $m_k(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\beta}_k, \boldsymbol{q}_2)$ (i.e., Equation (3.5)) as the metamodel objective function of iteration k and $h(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{q}_2) = 0$ (i.e., Equation (3.6)) be the constraints corresponding to the metamodel. We select U dimension by solving the following analytical optimization problem

$$\max_{\boldsymbol{x},\boldsymbol{z},\boldsymbol{u}} \quad m_k(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\beta}_k, \boldsymbol{q}_2) + \left(\sum_{i \in \mathcal{I}} \varepsilon u_i\right) \tag{3.8}$$

subject to

$$h(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{q}_2) = 0, \tag{3.9}$$

$$\boldsymbol{x} \in \mathcal{F}, \tag{3.10}$$

$$LB_i^{k-1} u_i + x_i^{k-1}(1 - u_i) \le x_i \quad \forall i \in \mathcal{I}, \tag{3.11}$$

$$x_i \le UB_i^{k-1} u_i + x_i^{k-1}(1 - u_i) \quad \forall i \in \mathcal{I}, \tag{3.12}$$

$$\sum_{i \in \mathcal{I}} u_i \le U, \tag{3.13}$$

$$u_i \in \{0, 1\}, \quad \forall i \in \mathcal{I}. \tag{3.14}$$

Constraint (3.9) forces the $\boldsymbol{x}$ to be within the problem feasible region, and (3.10) is the same to Constraint (3.6). Constraint (3.11) and (3.12) mean that when $u_i = 0$, x_i must be equal to x_i^{k-1}, and when $u_i = 1$, x_i can take value between LB_i^{k-1} and UB_i^{k-1}. Constraint (3.11) and (3.12) ensure that only the value of the selected dimensions are allowed to change, while for dimensions that are not selected, the values will be fixed to the current value of the current iterate. Constraint 3.13 ensures that at most U dimensions can be selected. Constraint 3.14 states that all u_i's must be binary. The term $\left(\sum_{i \in \mathcal{I}} \varepsilon u_i\right)$ in the objective function (3.8) allows us to select as more dimensions as we can. Since the value of ε is set to be close to zero, we are essentially maximizing the metamodel objective $m_k(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\beta}_k, \boldsymbol{q}_2)$ (i.e., Equation (3.5))

with additional requirement that at most U decision variables can be different from the current iterate $\boldsymbol{x}_k$.

The MIP (3.8)-(3.14) allows us optimally select U dimensions. We denote this procedure as *OptDim*. Solving it gives us $\boldsymbol{u}^k$, the optimal value of $\boldsymbol{u}$ of iteration k. We then use $\boldsymbol{u}^k$ to build a hyperbox

$$\mathcal{H}_k^{\text{opt}} = \left\{\boldsymbol{x} : LB_i^{k-1}u_i^k + x_i^{k-1}(1-u_i^k) \le x_i \le UB_i^{k-1}u_i^k + x_i^{k-1}(1-u_i^k), \forall i \in \mathcal{I}\right\}, \quad (3.15)$$

where u_i^k is the ith element of $\boldsymbol{u}^k$. In order to speed up the solution procedure, we may relax $\boldsymbol{x}$ to be continuous. Since the two metamodel we proposed in Section 3.2.3 are both MIPs, this dimension-selection problem is also a MIP.

With such a hyperbox which allows only U selected dimensions to change, in iteration k, we can sample from the lower-dimensional subregion $\mathcal{H}_k^{\text{opt}} \cap \mathcal{R}^{k-1}$ and thus reduce the size of the searching space. Also, these solved U dimensions are the result of optimizing a metamodel which approximates the simulation objective and accounts for the network effect such that customers may spillback from one station to its neighboring stations when there is no available car at the desired location.

3.2.3 Two MIPs for car-sharing network design

We present two MIPs that approximate the simulated estimation of the SO problem's objective function g (i.e. Equation (3.1)). Both of them are simplification of the simulator, and contain problem specific information. One MIP contains more detailed information about joint demand distribution over both time and space, and we call it *detailed model*. The other MIP contains less detailed information about demand distribution over space and time, and we denote it as *simple model*. With the problem specific information in the MIP, the simulator is no long a black box. Rather, it can be viewed as a "gray" box. The level of the problem specific information contained in the metamodel is an indicator of the gray level of the simulator, i.e., how much we know about the simulator. In Section 3.3, we will first use small-scale toy networks to compare how well the two metamodels can approximate the simulator. Then we

will study the performance of the algorithm, which we will propose in Section 3.2.4, with the two metamodels.

Detailed MIP

For the detailed MIP, we use the same one proposed in Section 2.2.3. We use the following notation.

- d_{tl}^{i}: number of customers that desire a reservation at station i with start time t and duration l;
- r_{tl}: revenue from a reservation with start time t and duration l;
- p^{ij}: discount to the revenue if a reservation is desired for station i but is fulfilled at (i.e., is made at) station j;
- z_{tl}^{i}: number of customers that make a reservation at station i with start time t and duration l;
- z_{tl}^{ij}: number of customers that desire to make a reservation at station i with start time t and duration l but make an adjusted reservation at station j with start time t and duration l;
- $t_{\max}$: number of one-hour reservation start time intervals during the planning period (e.g., for an n-day planning period, $t_{\max} = n \times 24$);
- $l_{\max}$: maximum reservation duration;
- $\mathcal{I}_i$: set of stations "near" station i, including station i;
- $\mathcal{L}$: set of reservation durations (in hours), $\mathcal{L} = \{1, 2, \ldots, l_{\max}\}$;
- $\mathcal{T}$: set of reservation start time interval indices, $\mathcal{T} = \{1, 2, \ldots, t_{\max}\}$;

The MIP is formulated as follows.

$$g_A^c(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} \sum_{t \in \mathcal{T}} \sum_{l \in \mathcal{L}} p^{ij} r_{tl} z_{tl}^{ij} - \sum_{i \in \mathcal{I}} c_i x_i. \tag{3.16}$$

The metamodel constraint h, i.e., Equation (3.6) consists of the Constraints (3.17)-(3.21).

$$\sum_{j \in \mathcal{I}_i} z_{tl}^{ji} = z_{tl}^{i} \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{3.17}$$

$$\sum_{j \in \mathcal{I}_i} z_{tl}^{ij} \leq d_{tl}^{i} \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{3.18}$$

$$\sum_{l \in \mathcal{L}} z_{tl}^{i} + \sum_{l \in \mathcal{L}} \sum_{t' \in \mathcal{T}_1(t,l)} z_{t'l}^{i} \leq x_i \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{3.19}$$

$$z_{tl}^{i} \in \mathbb{R}_+ \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{3.20}$$

$$z_{tl}^{ij} \in \mathbb{R}_+ \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}_i, \forall t \in \mathcal{T}, \forall l \in \mathcal{L}, \tag{3.21}$$

$$x \in \mathcal{F}, \tag{3.22}$$

where $\mathcal{T}_1(t,l) = \{t' \in \mathcal{T} : t' + 1 \leq t \leq t' + l - 1\}$. The objective function (3.16) is modeled as the the total revenue minus the cost. The cost part is the same as the SO problem, i.e., Equation (3.1). The revenue part sums up the revenues brought by all customers that have been served. The discount parameter, p^{ij}, are used to account for the customer lost and customer spillover in simulator when they cannot find available cars. Detail of customer spillback and spillover can be found in Fields et al. (2017). Constraint (3.17) is the customer flow conservation constraint. It shows that the z_{tl}^{i}, the total number of customers who want to user car at time t with duration l and actually use car at location i, is equal to the sum of customers who want to use car at time t with duration l at both i and nearby locations ($\forall j \in \mathcal{I}_i$). In Constraint (3.18), the left-hand-side is the total number of customers who want to user a car at time t location i for duration l and actually be served by the system; the right-hand-side is the total number of customers who want to user a car at time t location i for duration l. It means that the demand is higher than the number of customers served. Constraint (3.19) means that at any time and any location, the number of cars that are being used (i.e., $\sum_{l \in \mathcal{L}} \sum_{t' \in \mathcal{T}_1(t,l)} z_{t'l}^{i}$) and are about to be checked out by customers (i.e., $\sum_{l \in \mathcal{L}} z_{tl}^{i}$) should be smaller than the number of cars assigned to that location (i.e., x_i). Constraint (3.20) and (3.21) indicate that the auxiliary variables (i.e., z_{tl}^{i}'s and

z_{tl}^{ij}'s) are non-negative real numbers. We do not set them to integers, as we hope the metamodel can be solved fast. In this model, the auxiliary variable vector $\boldsymbol{z}$ consists of all of the z_{tl}^{i}'s and z_{tl}^{ij}'s. Finally, Constraint (3.22) means that the network design should be feasible, i.e., the Constraint (3.2)-(3.4) should be satisfied.

To estimate the demand parameter d_{tl}^{i}, we run the simulator for one replication with a network design such that all stations have unlimited number of cars. As a customer will always find an available car at the desired location and time, no customer spillover and customer lost will happen. Compared to the simulator, the MIP (3.16)-(3.21) are simplified in the following ways. First, the first-come-first-served principle is not kept. We include no information about reservation creation time in the model. Rather than forcing reservations that comes first to be fulfilled, the analytical model will choose a group of reservations that can result in the highest profit. Second, in the simulator, the customers can change both location and time if there are no available cars, while the analytical model only allows the change of location but not allow the change of time when customer spillback. Third, the analytical model uses the revenue discount parameter p^{ij} to approximate the spillback effect, while the simulator will try reservations that are similar (nearby location and slightly different time). Forth, the time resolution is 30 minute (i.e., both the reservation start time and duration can only increase for multiples of 30 minutes) in the simulator and 1 hour in the analytical model.

Simple MIP

To formulate the simple MIP, we use the following additional notation.

- d^i: total number of desired trips at station $i \in \mathcal{I}$
- r_i: average revenue of desired reservation at station i
- z^{ij}: number of customers that desire to make a reservation at station i with but make an adjusted reservation at station j
- t_i: average reservation duration of station i

The simple metamodel is formulated as follows.

$$g_A^s(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{q}_2) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} p^{ij} r_j z^{ij} - \sum_{i \in \mathcal{I}} c_i x_i \tag{3.23}$$

The MIP constraint h (i.e. Equation (3.6)) consists of Constraints (3.24)-(3.26).

$$\sum_{j \in \mathcal{I}_i} z^{ij} \leq d^i, \qquad \forall i \in \mathcal{I}, \tag{3.24}$$

$$\sum_{j \in \mathcal{I}_i} t_j z^{ji} \leq t_{\max} x_i \qquad \forall i \in \mathcal{I}, \tag{3.25}$$

$$z^{ij} \in \mathbb{R}^+, \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}_i, \tag{3.26}$$

$$\boldsymbol{x} \in \mathcal{F}. \tag{3.27}$$

Similar to the detailed MIP, the simple MIP's objective function (3.23) is computed as the revenue minus the cost. For the revenue part, we assume we only have the information about the average revenue of the successful reservation for each station. The total revenue, $\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} p^{ij} r_j z^{ij}$, is computed as the sum of revenues from all served customers. The revenue from each customer is base on the actual station the customer uses rather than the desired station. The total cost is computed in the same way as the SO problem (3.1) and the detailed model (3.16). Constraint (3.24) is the demand constraint, which means that the number of served customers should be smaller than the demand. Constraint (3.25) is the supply constraint. It means that the total reservation hours used by one location should be smaller than the total available reservation hours, which is simply the number of one-hour reservation start time intervals ($t_{\max}$) times the number of cars assign to that location. Constraint (3.26) require the auxiliary variables (z^{ij}'s) be non-negative real numbers. In this model, the auxiliary variable vector $\boldsymbol{z}$ consists of all of the z^{ij}'s. Finally, Constraint (3.27) indicates that the network design should be feasible.

In the simple MIP, we run the simulator for one replication with a network design such that all stations have unlimited number of cars to estimate the demand parameter d^i's, the revenue parameter r_i's reservation duration parameter t_i's. We

count the number of sampled reservation for each station. Compared to the detailed model (3.16)-(3.21), the simple model has no detailed information about the demand distribution over time. Only average reservation duration is given for all stations (i.e., t_i's). No information about reservation start time is provided. Also, the simple MIP still track the customer spillback, but in a aggregated way such that customers with different start time and duration are not distinguished. Compared to the simulator of Fields et al. (2017), the simple MIP inherits the detailed model's features: the first-come-first-reserve principle are not enforced, the customers are allowed only to change the location, and the spillback effect is represented by the revenue discount parameter p^{ij}.

3.2.4 Algorithm

In this section, we propose a globally convergent discrete SO algorithm MetaESBB-OptDim. It is an extension of Empirical Stochastic Branch-and-Bound (ESBB) (Xu and Nelson, 2013). The main idea of ESBB is to iteratively partition the feasible region of the problem into smaller subregions, and focus on sampling from more potential subregions. One iteration contains three parts: partitioning, bounding and updating. In each iteration, ESBB first partitions current best subregion into several non-overlapping and non-empty subregions. Then it randomly samples a certain number of solutions from the newly created subregions, and also randomly sample some solutions from subregions generated in previous iterations. Next, these solutions are simulated and their mean objective values are computed. The subregion containing the current iterate (i.e., the current best solution) is considered as the best subregion. The best solution is the one with the best mean simulated objective value. Our extension of ESBB maintains the property that all feasible solutions have a positive probability to be sampled and simulated in every iteration, hence maintains the globally convergent property. For more details about ESBB, please refer to Xu and Nelson (2013).

We use the following addition notation.

- $\mathcal{G}^k$: all feasible solutions that have been simulated at least one replication at the end of iteration k.
- $\boldsymbol{x}^k$: the current iterate, a solution in $\mathcal{G}^k$ with the best mean simulated objective value.
- $\mathcal{P}^k$: the partition of $\mathcal{F}$ at iteration k, the set of all subregions.
- $\mathcal{P}(\mathcal{R})$: set of newly created subregions that are partitioned from $\mathcal{R}$.
- $\mathcal{S}^k$: set of solutions to be simulated in iteration k.
- v_R: total number of solutions to be sampled from newly created subregions in $\mathcal{P}(\mathcal{R}^{k-1})$ in each iteration during iteration k.
- v_O: total number of solutions to be sampled from existing subregions in $\mathcal{P}^k \setminus \mathcal{P}(\mathcal{R}^{k-1})$ during iteration k.
- v_L: number of solutions to be sampled from the lower-dimensional subregion in each iteration.
- Δn_F: number of simulation runs to conduct for a solution if it has never been simulated in previous iterations.
- Δn_A: number of simulation runs to conduct for a solution if it has been simulated in previous iterations.
- $\boldsymbol{x}_k^{\text{meta}}$: solution obtained by solving the metamodel in iteration k.
- $\boldsymbol{x}_k^{\text{meta-sub}}$: solution obtained by solving the metamodel in iteration k, with additional constraint that the solution must be in $\mathcal{R}^{k-1}$.
- K: the maximum number of iterations to run.

We propose MetaESBB-OptDim, an algorithm which combines metamodel and ESBB, and present it in Algorithm 2. MetaESBB-OptDim contains the following steps. At the beginning, we set the value for the related parameters and initialize the

algorithm. In Step 1, we partition the feasible region. In Step 2, we obtain solutions that needed to be simulated. Step 2a means that the current iterate will always be simulated for additional replications since the second iteration. Solutions can be sampled from both newly generated subregions and subregions generated in previous solutions. In Step 2b, we sample in total v_R solutions from newly generated partitions. In Step 2c, we allocate the v_O sampling budget using the upper confidence bound Chebyshev (UCB-Chebyshev) method proposed in Xu and Nelson (2013). Same as MetaAHA, we use the RMD algorithm of Hong and Nelson (2006) to sample solutions. RMD sampling is an asymptotically uniform sampling strategy. This method samples integral points from a compact set. We obtain solutions by solving the metamodel in Step 2d. We first solve the fitted metamodel to obtain one solution (i.e., $\boldsymbol{x}_k^{\text{meta}}$), and solve it again with additional constraints forcing the solution to be within, the best subregion identified in the previous iteration to obtain another solution (i.e., $\boldsymbol{x}_k^{\text{meta-sub}}$). At Step 2e, MetaESBB-OptDim will solve the dimension-selection problem to generate the hyperbox $\mathcal{H}_k^{\text{opt}}$, and then randomly sample solutions from the lower-dimensional subregion $\mathcal{H}_k^{\text{opt}} \cap \mathcal{R}^{k-1}$. In Step 3, we simulate the solutions obtained in Step 2. In Step 4, we update the current iterate and current best subregion. In Step 5, if we reach the maximum iteration number, we stop and return the current iteration. Otherwise, we compute the analytical objective value for solutions, update parameters of the metamodel and the iteration counter, and proceed to the next iteration.

We use metamodel for two purposes: (1) we solve the fitted metamodel in each iteration to obtain new solutions as discussed in Section 3.2.1 (2) we use the metamodel to build lower-dimensional subregions to sample solutions as discussed in Section 3.2.2. Algorithm 2 is essentially ESBB if we remove Step 2d, Step 2e, Step 5b and Step 5c.

In order to guarantee global convergence, as stated in Xu and Nelson (2013), during iteration k ($k > 0$), the union of all subregions in $\mathcal{P}_k$ should contain all feasible solutions, and all subregions should be non-overlapping. In other words, no matter what partition method we use, we should guarantee that $\bigcup_{\mathcal{R}\in\mathcal{P}_k} = \mathcal{F}$ and $\bigcap_{\mathcal{R}\in\mathcal{P}_k} = \varnothing$. This indicates that whenever we partition a subregion $\mathcal{R}$, we should also guarantee

Algorithm 2 MetaESBB-OptDim

Initialization:

- Initialize parameters: set values for K, v_R, v_O, Δn_F, Δn_A; let $\mathcal{R}^0 = \mathcal{F}$ and $\mathcal{P}^0 = \{\mathcal{F}\}$; let iteration index $k = 1$.

Step 1: partitioning

- Partition $\mathcal{R}^{k-1}$ to create $\mathcal{P}(\mathcal{R}^{k-1})$, and let $\mathcal{P}^k = (\mathcal{P}^{k-1} \setminus \{\mathcal{R}^{k-1}\}) \cup \mathcal{P}(\mathcal{R}^{k-1})$.

Step 2: sampling

- Step 2a: if $k = 1$, let $\mathcal{S}^k = \varnothing$; if $k > 1$, $\mathcal{S}^k = \{\boldsymbol{x}^{k-1}\}$.
- Step 2b: randomly sample v_R solutions from the newly create partitions; add all sampled solutions to $\mathcal{S}^k$.
- Step 2c: if $k = 1$, do nothing; if $k > 1$, randomly sample v_O solutions from old subregions in $\mathcal{P}_{k-1} \setminus \{\mathcal{R}^{k-1}\}$.
- Step 2d: solve the metamodel (3.5)-(3.7) to obtain $\boldsymbol{x}_k^{\text{meta}}$; if $k > 1$, solve the metamodel (3.5)-(3.7) with extra constraint $\boldsymbol{x} \in \mathcal{R}^{k-1}$ to obtain $\boldsymbol{x}_k^{\text{meta-sub}}$; add $\boldsymbol{x}_k^{\text{meta}}$ and $\boldsymbol{x}_k^{\text{meta-sub}}$ to $\mathcal{S}^k$.
- Step 2e: if $k = 1$, do nothing; if $k > 1$, solve the dimension-selection problem (3.8)-(3.14) and build $\mathcal{H}_k^{\text{opt}}$, randomly sample v_L solutions from the lower-dimensional subregion $\mathcal{H}_k^{\text{opt}} \cap \mathcal{R}^{k-1}$, then add them to $\mathcal{S}^k$.
- Step 2f: if $k = 1$, let $\mathcal{G}^1 = \mathcal{S}^1$; if $k > 1$, let $\mathcal{G}^k = \mathcal{G}^{k-1} \cup \mathcal{S}^k$.

Step 3: simulation

- For each solution in $\mathcal{S}^k$: simulate Δn_F replications if it has never been simulated before; otherwise, simulate Δn_A replications.

Step 4: update current iteration and best subregion

- Step 4a: let $\boldsymbol{x}^k$ be the solution in $\mathcal{G}^k$ with the best average simulated objective value.
- Step 4b: let $\mathcal{R}^k$ be the subregion in $\mathcal{P}^k$ that contains $\boldsymbol{x}^k$.

Step 5: stop or update metamodel and iteration counter

- Step 5a: if $k = K$, terminate and return $\boldsymbol{x}^k$; otherwise, set $k = k + 1$.
- Step 5b: for any simulated point $\boldsymbol{x}$ that has not been evaluated by the analytical network model, evaluate it (i.e., for a given $\boldsymbol{x}$, maximize $g_A(\boldsymbol{x}, \boldsymbol{z})$ over $\boldsymbol{z}$ subject to Constraints (3.6)).
- Step 5c: use all simulation observations collected so far to fit the metamodel parameter $\boldsymbol{\beta}_k$ (i.e., solve the least squares Problem (A.1) defined in Appendix A).
- Step 5d: proceed to Step 1.

that newly created subregions should cover all parts of $\mathcal{R}$, while each of these new subregions should share no common element, i.e., $\bigcup_{\mathcal{R}' \epsilon \mathcal{P}(\mathcal{R})} = \mathcal{R}$ and $\bigcap_{\mathcal{R} \epsilon \mathcal{P}(\mathcal{R})} = \varnothing$. We use a generic partition method that will keep the current iterate in the relatively interior part of one of the newly created subregions. See Algorithm 4 in Appendix E for details of this partition method.

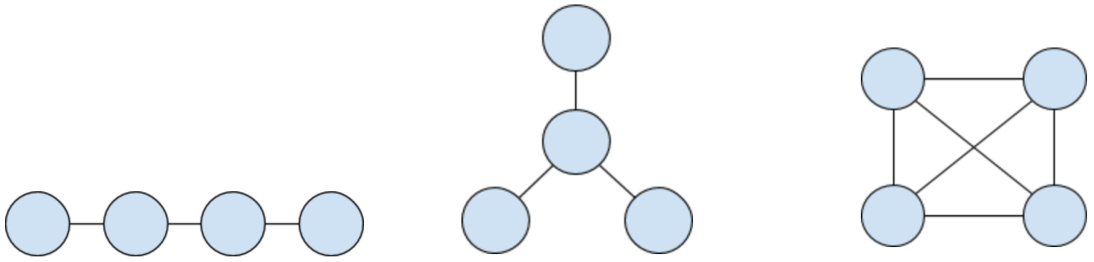
3.3 Case studies

In this section, we apply the propose algorithms to several case studies. We first study three two networks. Then we present the results of the comparison of the algorithms using two high-dimensional problem using data from Zipcar's Boston two-way car-sharing fleet. All experiments are conducted on a machine with 125GB RAM and an Intel Xeon E5-2630 v3 processor.

3.3.1 Evaluation of metamodel accuracy

In this section, we use toy networks to evaluate the ability of both the simple and detailed metamodels to approximate the simulated objective value. We use three toy networks that are presented in Figure 3-2. All three toy networks have four stations. Each circle represent one station. If two circles are connected, customers who wants to use car-sharing service at one station may spillback to the other one. Otherwise, customers will not spillback to the other station. Figure 3-2a shows a tandem network, where customers from one station can only spillback to the left or right neighboring stations. Figure 3-2b shows a centralized network, where one station may have its customers spillback to all other three stations, and customers from the other three stations can only spillback to the center station. Figure 3-2c shows a fully connected network, where customers from any station can spillback to all other three stations. For each network, all stations have capacity of 6 (i.e., $N^i = 6$). The fleet size is unlimited (i.e., $X \geq 24$). Hence, for each toy network, there are in total 2401 feasible solutions (i.e., $|\mathcal{F}| = 2401$).

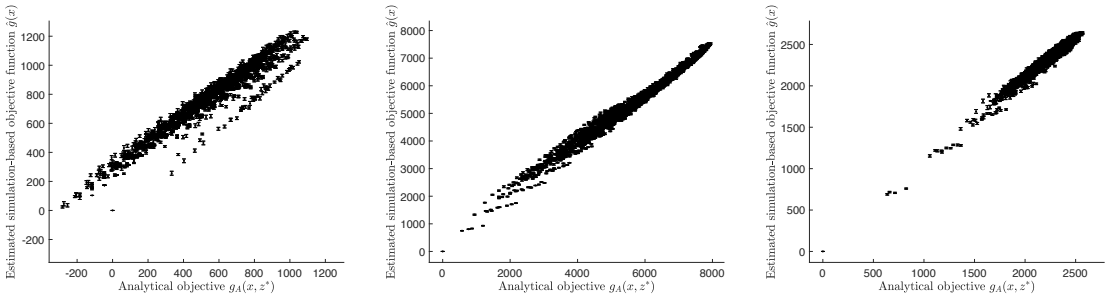
For all feasible solutions in each network, we compare the analytical objective



(a) Toy network 1 (b) Toy network 2 (c) Toy network 3

Figure 3-2: Toy network topologies

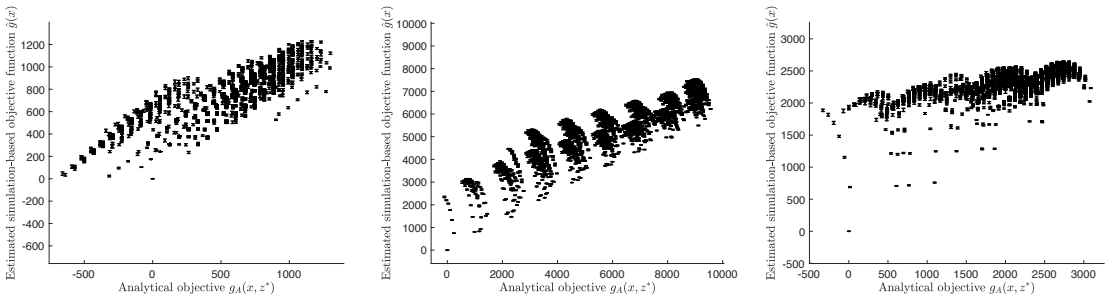
value and simulated objective value. For a given feasible solution $\bar{\boldsymbol{x}} \in \mathcal{F}$, its analytical objective is computed by maximizing the metamodel objective function over auxiliary variable $\boldsymbol{z}$ with $\boldsymbol{x} = \bar{\boldsymbol{x}}$ in problem (3.5)-(3.7). Specifically, $\max_{\boldsymbol{z}} g_A^c(\bar{\boldsymbol{x}}; \boldsymbol{z})$ subjected to Constraints (3.17) -(3.21) for the detailed metamodel and $\max_{\boldsymbol{z}} g_A^s(\bar{\boldsymbol{x}}; \boldsymbol{z})$ subjected to Constraints (3.24)-(3.26) for the simple metamodel. For the simulated objective function, we run 20 simulation replications for each solution to estimate objective function and to compute the 95% confidence intervals.



(a) Toy network 1 (b) Toy network 2 (c) Toy network 3

Figure 3-3: Objective function values: analytical vs. simulated (detailed metamodel)

We plot the results for the result for the detailed metamodel in Figure 3-3 and the simple metamodel in Figure 3-4. Each figure contains three subfigures, and each subfigure corresponds to one toy network. The confidence intervals are very small and are barely visible. In each subfigure, the x-axis is the value for analytical objective function, and the y-axis is the value for simulated objective function. We can see that the detailed metamodel shows a better approximation for the simulated objective values, compared to the simple metamodel. We compute the Pearson correlation coefficient



(a) Toy network 1 (b) Toy network 2 (c) Toy network 3

Figure 3-4: Objective function values: analytical vs. simulated (simple metamodel)

between the analytical objective values and the mean simulated objective values of all 2401 feasible solutions for each toy network. The Pearson correlation coefficients for the detailed metamodel are 0.974 (toy network 1), 0.992 (toy network 2) and 0.979 (toy network 3). For the simple metamodel, Pearson correlation coefficients are 0.847 (toy network 1), 0.881 (toy network 2) and 0.645 (toy network 3). The comparison of the Pearson correlation coefficients between the two metamodels shows that the detailed metamodel is a better approximation of the simulated objective function compared to the simple metamodel.

3.3.2 High-dimensional case study with 144 stations

In this section, we study the performance of the proposed algorithm using a case with 144 stations. These stations are located in the cities of Cambridge, Somerville and Medford of Massachusetts, USA. The map in Figure 3-5 shows the location of these 144 stations. Section 2.3.4 showed that the accuracy of the detailed metamodel approximating the simulated objective function decreases as the demand and cost level increases. To better demonstrate the performance of the propose algorithms, in this section, we increase the demand level to 3 times of the trained demand for each station and the cost level to 3 times of the real cost level. For more about the trained demand, see the Section 2 of Fields et al. (2018).

For experiments in this section, we run 10 simulation replications for newly obtained solutions and 2 simulation replications if a solution has already been simulated (i.e, $\Delta n_F = 10$ and $\Delta n_A = 2$). When building a lower-dimensional subregion (either

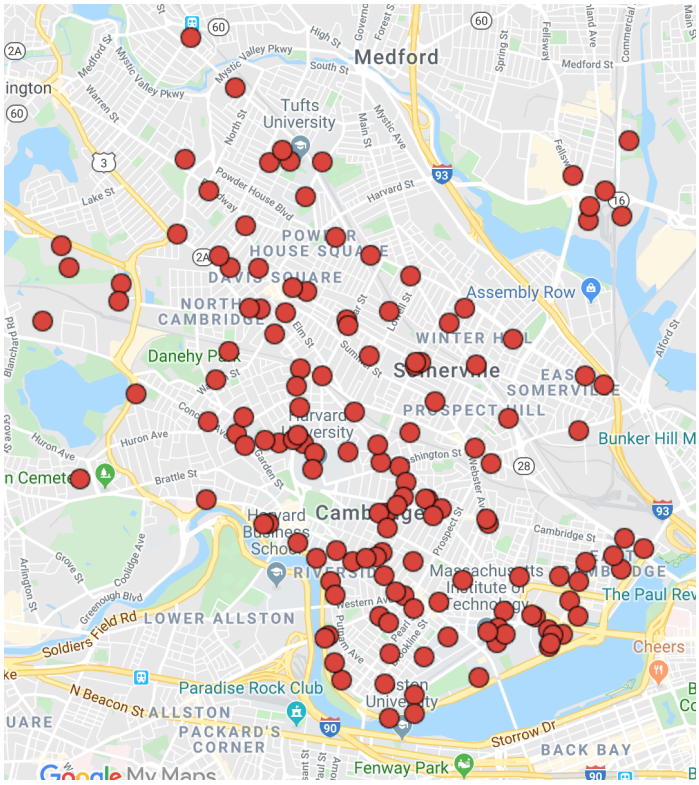


Figure 3-5: Zipcar stations in Boston South End neighborhood (map data: Google Maps (2019))

optimally or randomly), we select 5 dimensions (i.e., $U = 5$).

Added value of using MIP for algorithm initialization

In this section, we compare ESBB algorithm with different initial solutions obtained by solving the simple metamodel and detailed metamodel. No lower-dimensional subregion will be created. We run three algorithms: (1) ESBB of Xu and Nelson (2013); (2) ESBBInit-Detailed, which is ESBB with one initial solution as the solution of the unfitted detailed metamodel (i.e, maximize $g_A^c(\boldsymbol{x}; \boldsymbol{z})$ subjected to Constraints (3.17) -(3.22)); (3) ESBBInit-Simple, which is ESBB with one initial solution as the solution of the unfitted simple metamodel (i.e., maximize $g_A^s(\boldsymbol{x}; \boldsymbol{z})$ subjected to Constraints (3.24)-(3.27)). We set $v_R = 25$, and $v_O = 4$. Hence, there will be 29 solutions simulated in each iteration. For all algorithm runs, we run each for 80 iterations (i.e., $K = 80$)

In Figure 3-6, we present the result of the three algorithms. The x-axis is the

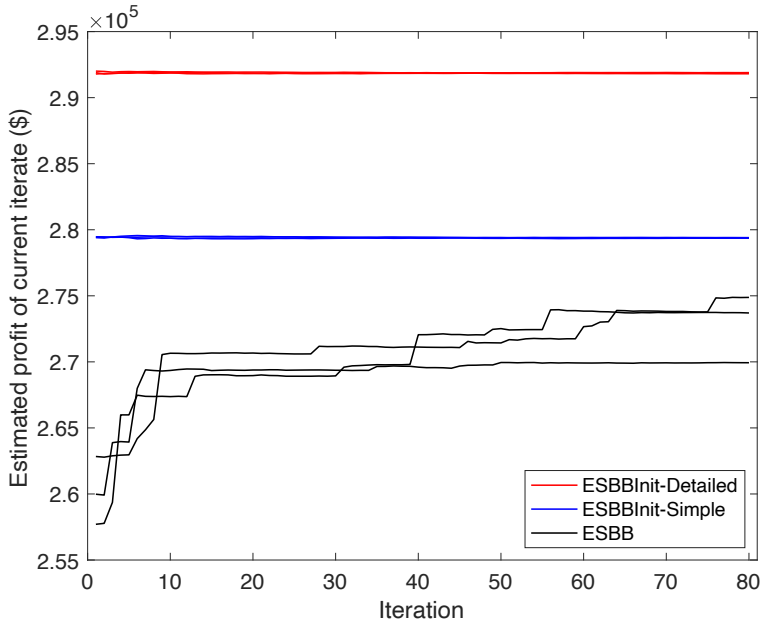


Figure 3-6: ESBB with different initial solutions

iteration index, and the y-axis the the simulated mean objective value of the current iterate. we run each algorithm 3 times. The black lines are for ESBB, the blue lines are for ESBBInit-Simple, and the red lines are for ESBBInit-Detailed. The red lines start with the solution of the detailed metamodel, and have not improved since the first iteration. The blue lines start with the solution of the simple metamodel and also show no improvement since the first iteration. Such results shows that a metamodel with problem specific information has the potential to help ESBB find better solutions. But we need more than just initializing the ESBB with metamodel solutions.

Added value of using MIP information for subregion identification

We have demonstrate the added value of using the MIP for to initialize ESBB. Here, we consider using the MIP information only for subregion identification. Specifically, we propose ESBBInit-OptDim, which builds the lower-dimensional by solving the dimension-selection problem (3.8)-(3.14). ESBBInit-OptDim is essentially MetaESBB-OptDim without solving the updated metamodel for solutions in each iterate, i.e., Algorithm 2 without Step 2d. For a benchmark purpose, we also pro-

pose ESBBInit-RanDim, which builds the lower-dimensional by randomly select U dimensions. It is essentially Algorithm 2 without Step 2d, 5b and 5c, and build lower-dimensional subregion in Step 2e using the method described in Appendix F. Both ESBBInit-OptDim and ESBBInit-RanDim are initialize by the solution of the unfitted metamodel.

In Figure 3-7, using the simple metamodel, we compare ESBBInit-OptDim and ESBBInit-RanDim. We run each algorithm 6 times. For both methods, we set K = 80, v_R = 14, v_L = 10 and v_O = 4. Plus the current iterate, in each iteration, we sample in total 29 solutions. The figure format is the same as Figure 3-12. The red line and red shaded area represent ESBBInit-OptDim, and the blue line and blue shaded area represent ESBBInit-RanDim. For all 6 runs of the two methods, all current iterate except for the first iteration are identified by sampling from the lower-dimensional subregion. The first iteration current iterate are the solution obtained by solving the unfitted simple metamodel. ESBBInit-OptDim outperforms ESBBInit-RanDim. Note that the time spent on building lower-dimensional subregion using the simple metamodel is negligible, compare to the time for simulation. This indicate that sampling from the lower-dimensional subregion built by solving the dimension-selection problem using metamodel can result in solutions with better performance than sampling from randomly built lower-dimensional subregions. Metamodel can guild the algorithm find better solutions by using it in the lower-dimensional subregion creation.

We also compare ESBBInit-OptDim and ESBBInit-RanDim using the detailed metamodel. We run each algorithm 6 times. For both methods, we set K = 80, v_R = 14, v_L = 10 and v_O = 4. Plus the current iterate, in each iteration, we sample in total 29 solutions. Figure 3-8, which has the same figure format as Figure 3-7, shows the change of the objective value as the iteration index increases. We see that the red shaded area quickly becomes higher than the blue shaded area. For all runs of the two algorithm, all current iterates are identified by sampling from the lower-dimensional subregion. This shows that given the same simulation budget, lower-dimensional subregion built by using the metamodel can provide better solutions than lower-

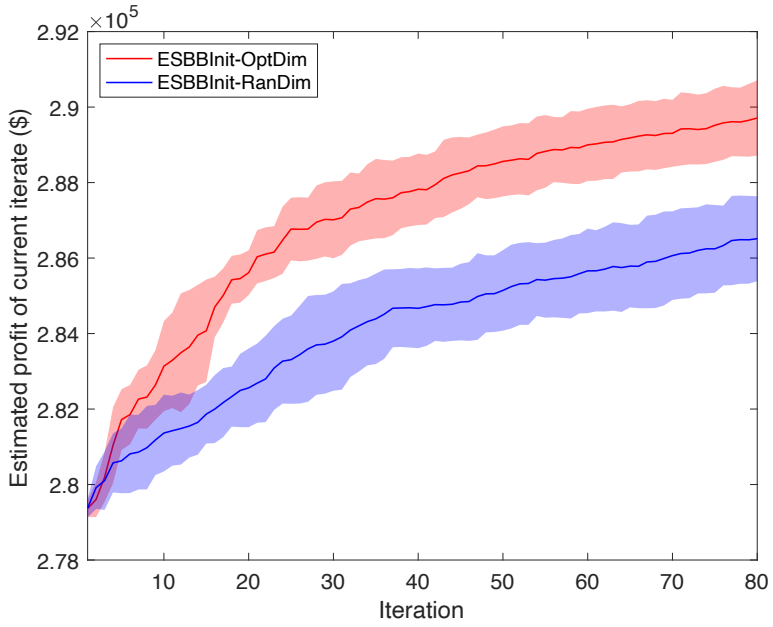


Figure 3-7: ESBBInit-OptDim vs. ESBBInit-RanDim (simple metamodel)

dimensional subregion built by random dimension-selection. This is consistent with the case of simple metamodel, as shown in Figure 3-7.

In Figure 3-9, we plot the change of objective values against elapsed time of all runs of ESBBInit-OptDim and ESBBInit-RanDim with the detailed metamodel. When all the ESBBInit-RanDim (blue lines) stops, There are five red lines are higher than all blue lines. This shows that ESBBInit-OptDim outperforms ESBBInit-RanDim. We see that the red lines last longer than the blue lines. Note both algorithms run 80 iterations, the additional time of ESBBInit-OptDim comes from solving the dimension-selection problem using the detailed model. This indicate that when solving the metamodel-based dimension-selection time is not negligible compared to simulation, there may be a trade-off between solution quality and algorithm run time.

Added value of using MIP information for both metamodel optimization and subregion identification

In above sections, we show the benefit of using MIP information to build a lower-dimensional subregion for sampling. Previous researches such as Osorio and Bierlaire (2013), Zhang et al. (2017) and Zhou et al. (2019) (i.e., Chapter 2) show the benefit of

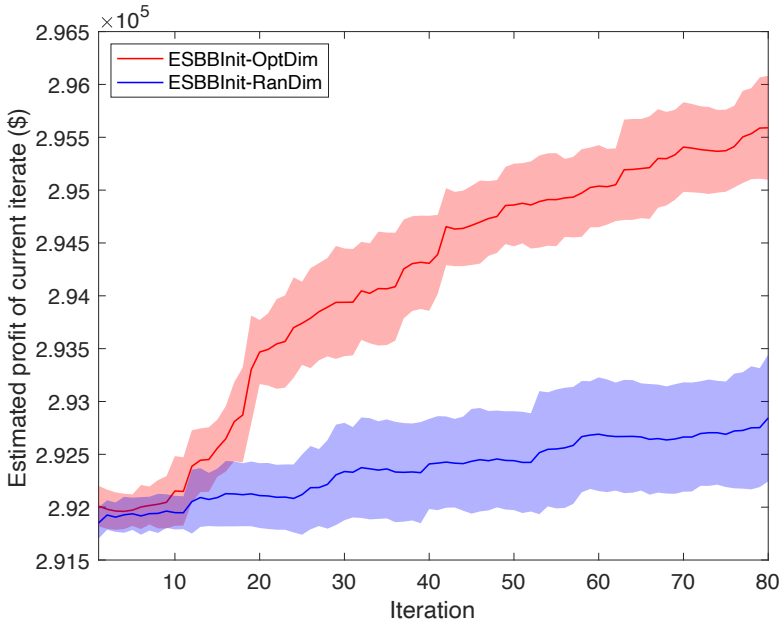


Figure 3-8: ESBBInit-OptDim vs. ESBBInit-RanDim (detailed metamodel)

solving the fitted metamodel to obtain solutions in each iteration. In this section, we study whether using the lower-dimensional subregion built by solving the dimension-selection problem (3.8)-(3.14) can bring additional benefit given that we solve the fitted metamodel to obtain solutions in each iteration

We first compare the MetaESBB-OptDim and MetaESBB. MetaESBB combines metamodel with ESBB. It only solves the metamodel for solutions but does not build lower-dimensional subregion for sample. It is essentially MetaESBB-OptDim (i.e., Algorithm 2) without Step 2e. We run MetaESBB-OptDim and MetaESBB each for 6 times using the simple metamodel and each for another 6 times using the detailed metamodel. For all 24 algorithm runs, we run each for 80 iterations (i.e., $K = 80$). For MetaESBB-OptDim, we set $v_R = 12$, $v_L = 10$, and $v_O = 4$. For MetaESBB, we set $v_R = 22$, and $v_O = 4$. Plus the two solutions obtained by solving the metamodel and the current iterate, 29 solutions will be simulated for both MetaESBB-OptDim and MetaESBB in each iteration.

In Figure 3-10, using the simple metamodel, we plot the comparison between MetaESBB-OptDim and MetaESBB. The x-axis is the iteration index, and the y-axis is the estimation of the objective value of the current iterate. The red line is

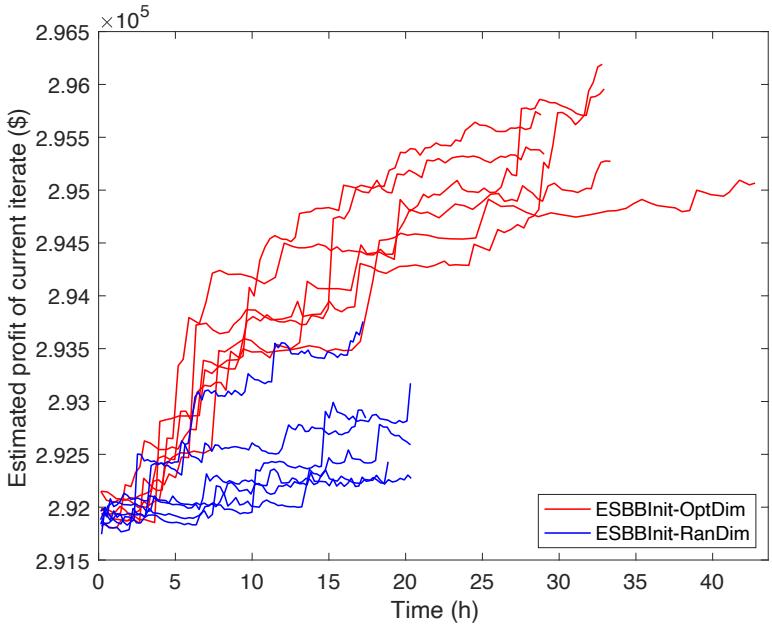


Figure 3-9: Change of current iterate objective value over time: ESBBInit-OptDim vs. ESBBInit-RanDim (detailed metamodel)

the average objective function value of the 6 run of MetaESBB-OptDim, i.e., for each iteration, we compute and plot the mean objective value of the 6 current iterates of the 6 runs of MetaESBB-OptDim. The red shaded area is the 95% confidence interval of mean objective values of the 6 runs of MetaESBB-OptDim, i.e., for each iteration, we compute and plot the 95% confidence interval of the mean objective value of the 6 current iterates. The black line is the average objective value of the 6 run of MetaESBB, with the gray shaded area as the 95% confidence interval. We see that the red line is higher than the black line. The red shaded area has almost no overlap with the black shaded area. In the case of using the simple metamodel for both optimization and lower-dimensional subregion building, the time spent on solving the dimension-selection problem (i.e., Step 2e of Algorithm 2) as well as solving the fitted metamodel for solution (i.e., Step 2d of Algorithm 2) are negligible compared to the simulation time in each iteration, so we only plot the change of objective values against iteration index for the simple metamodel.

We compare the performance of MetaESBB-OptDim and MetaESBB using the detailed metamodel in Figure 3-11. The figure format is the same as Figure 3-10. We

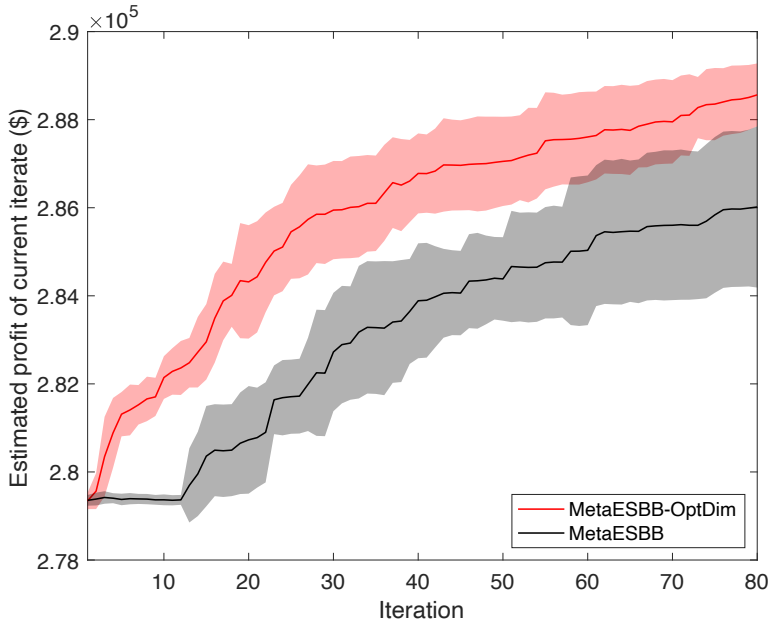


Figure 3-10: Added value of building lower-dimensional subregion using metamodel (simple metamodel)

can see that after the first several iterations, the red line is above the black curve, and the red shaded area have no overlap with the gray shaded area after iteration 16. Note that in Figure 3-11, the x-axis is the iteration index not the time. This means that with the same simulation budget, using the lower-dimensional subregion built by solving the dimension-selection problem needs fewer simulations to achieve better solutions than not using the lower-dimensional subregion.

Compared to the blue lines (ESBBInit-Simple) and red lines (ESBBInit-Detailed) in Figure 3-6, we note that the black lines in both Figure 3-10 (MetaESBB with simple metamodel) and 3-11 (MetaESBB with detailed metamodel) show an increase of objective value as the algorithm advances. This means that for both the simple and detailed metamodel, using the MIP information in each iteration is an effective way for discrete SO algorithms to identify better solutions after the first iteration. This is consistent with the findings in Section 2.3.2 and 2.3.3.

Table 3.1: Percentage of how current iterates are found in MetaESBB-OptDim

	Solving updated metamodel	Lower-dimensional subregion sampling
Simple metamodel	16%	84%
Detailed metamodel	77%	23%

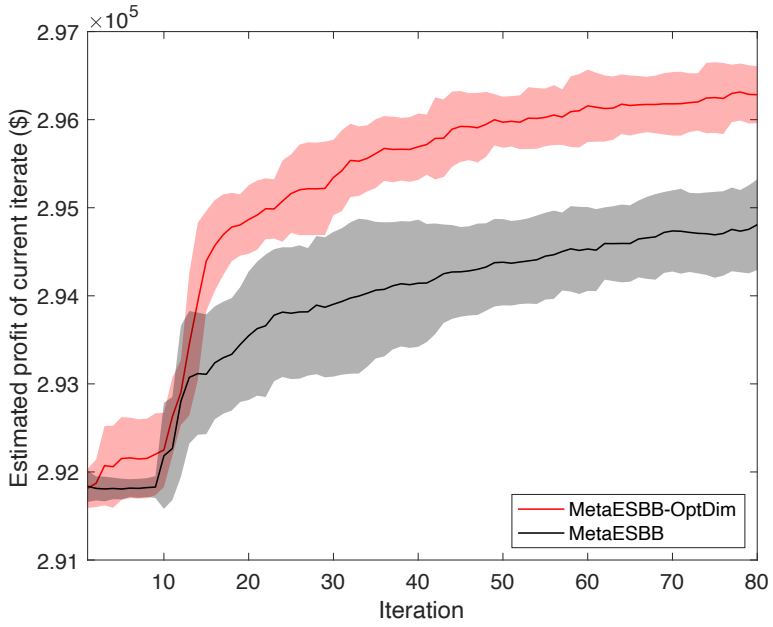


Figure 3-11: Added value of building lower-dimensional subregion using metamodel (detailed metamodel)

In terms of how current iterates are identified, for MetaESBB (using either simple or detailed metamodel), all current iterates are found by solving the updated metamodel. In Table 3.1, we summarize the percentage of how current iterates are found in MetaESBB-OptDim using the two metamodels. When we use the simple metamodel, most (84%) of current iterates are identified by sampling from lower-dimensional subregion. When we use the detailed metamodel, most (77%) current iterates are found by solving the updated metamodel. This may indicate that when using a more accurate metamodel, solving the fitted metamodel for solution is better than sampling from the lower-dimensional subregion, but otherwise when the metamodel is less accurate.

In Figure 3-12, using the simple metamodel, we benchmark MetaESBB-OptDim with MetaESBB-RanDim. MetaESBB-RanDim follows most steps of MetaESBB-OptDim except for how to build the lower-dimensional subregion (i.e., Step 2e of Algorithm 2). Instead of solving the dimension-selection problem (i.e. problem (3.8)-(3.14)), MetaESBB-RanDim builds the lower-dimensional subregion by randomly sampling dimensions. See Appendix F for details. We run MetaESBB-RanDim

for 6 times. For MetaESBB-RanDim, we set all algorithm control parameters the same as MetaESBB-OptDim. The figure axes the same as Figure 3-10. The red line correspond to the mean performance of MetaESBB-OptDim, and the red shaded area represent the 95% confidence interval. The blue line correspond to the mean performance of MetaESBB-RanDim, and the blue shaded area represent the 95% confidence interval. Both methods have similar performance. MetaESBB-OptDim is slightly better.

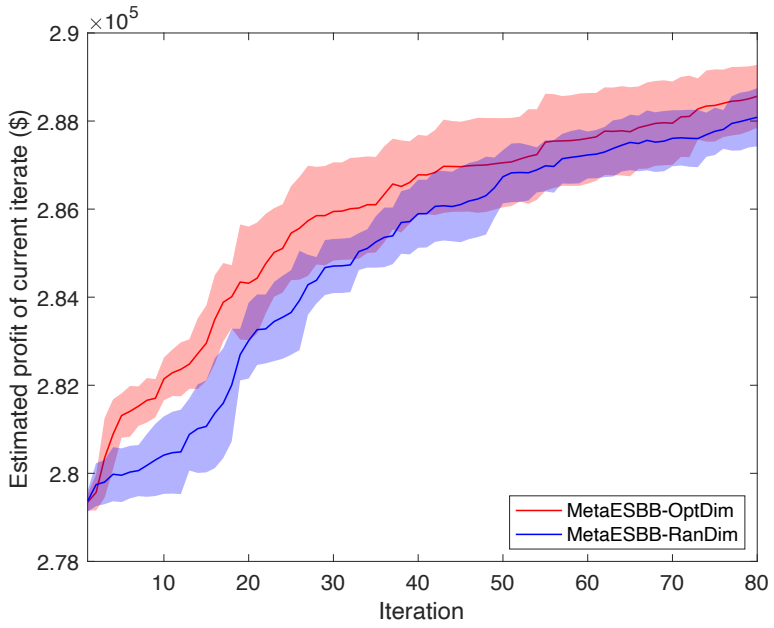


Figure 3-12: Benchmark random dimension-selection with dimension-selection using metamodel (simple metamodel)

In Figure 3-13, using the detailed metamodel, we benchmark MetaESBB-OptDim with MetaESBB-RanDim. We also run MetaESBB-RanDim for 6 times. For MetaESBB-RanDim, we set all algorithm control parameters the same as MetaESBB-OptDim. The figure format is the same as Figure 3-12. MetaESBB-OptDim which uses the metamodel to build the lower-dimensional subregion have better performance than MetaESBB-RanDim which randomly build the lower-dimensional subregion, given the same simulation budget.

In this case of 144 stations, the time for solving the dimension-selection problem using the detailed metamodel are not negligible compared to the time for simula-

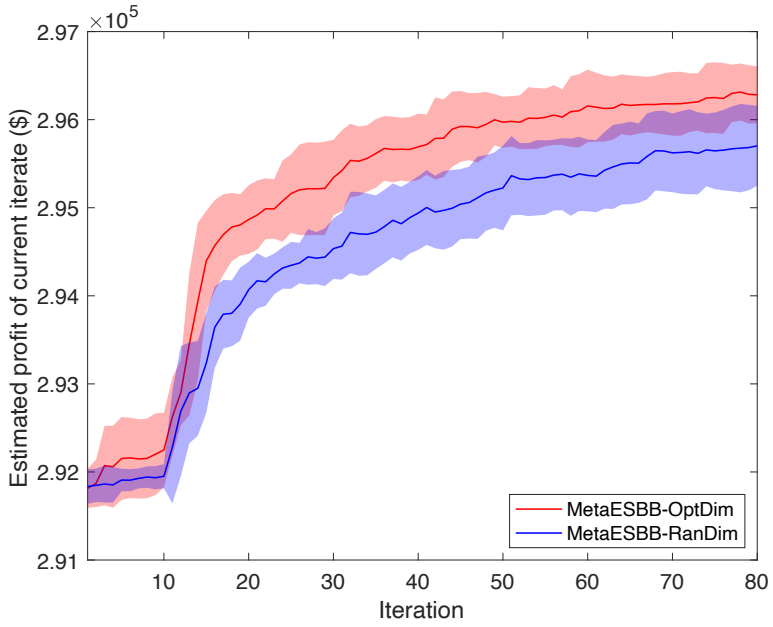


Figure 3-13: Benchmark random dimension-selection with dimension-selection using metamodel (detailed metamodel)

tion. In Figure 3-14, we plot the algorithm runs of MetaESBB-OptDim (red lines), MetaESBB-RanDim (blue lines) and MetaESBB (black lines) The x-axis is the time. The y-axis the simulated mean objective value of the current iterate. We can see that several red lines are longer. For the 6 runs of MetaESBB-OptDim using the detailed metamodel, the time for solving one dimension-selection problem is 180.1 seconds on average. The simulation time per iteration on average is 781.2 seconds. This also indicates a trade-off about time and algorithm performance.

In Table 3.2, we summarize the percentage of how current iterates are found in MetaESBB-RanDim using the two metamodels. When we use the simple metamodel, most (87%) of current iterates are identified by sampling from lower-dimensional subregion. When we use the detailed metamodel, most (72%) current iterates are found by solving the updated metamodel. These values are similar to the case of MetaESBB-OptDim (i.e., Table 3.1). This result indicate that randomly built lower-dimensional subregion has the potential to guide the algorithm to better solutions efficiently.

We run additional simulation replications to compare solutions obtained by MetaESBB-

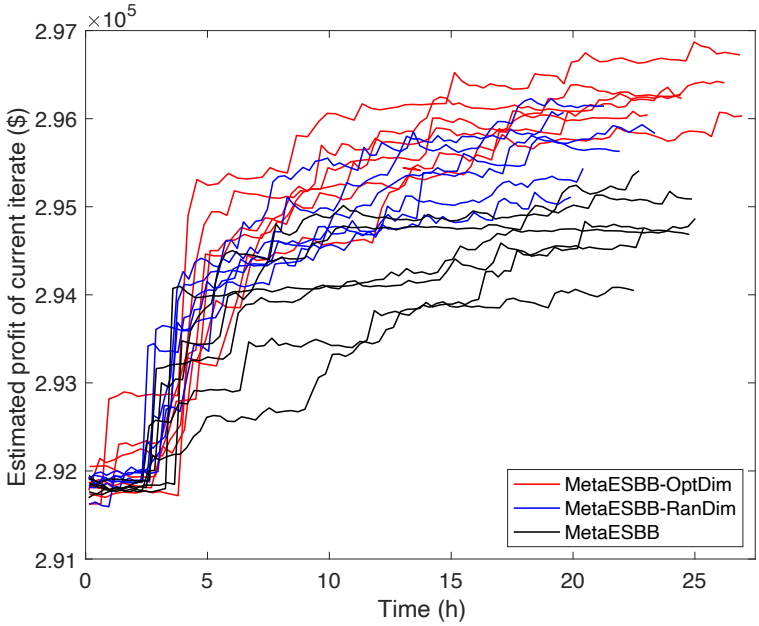


Figure 3-14: Change of current iterate objective value over time: MetaESBB-OptDim vs. MetaESBB-RanDim vs. MetaESBB (detailed metamodel)

Table 3.2: Percentage of how current iterates are found in MetaESBB-RanDim

	Solving updated metamodel	Lower-dimensional subregion sampling
Simple metamodel	13%	87%
Detailed metamodel	72%	28%

OptDim, MetaESBB and MetaESBB-RanDim using both the simple and detailed metamodel. Detailed results are summarized in Appendix G.

On when to use MIP information for either metamodel optimization or for subregion identification

In the above discussion, we use metamodel in the ESBB framework mainly for two purposes - solving for solutions in each iteration and building a lower-dimensional subregion for focused sampling. We also find that for MetaESBB-OptDim, more current iterate are found by sampling from the lower-dimensional subregion when using the simple metamodel, while more current iterate are found by solving the fitted metamodel when using the detailed metamodel. In Section 3.3.1, we use toy networks to show that the simple metamodel is a less accurate approximation for the simulator than the detailed metamodel. In this section, we study which method

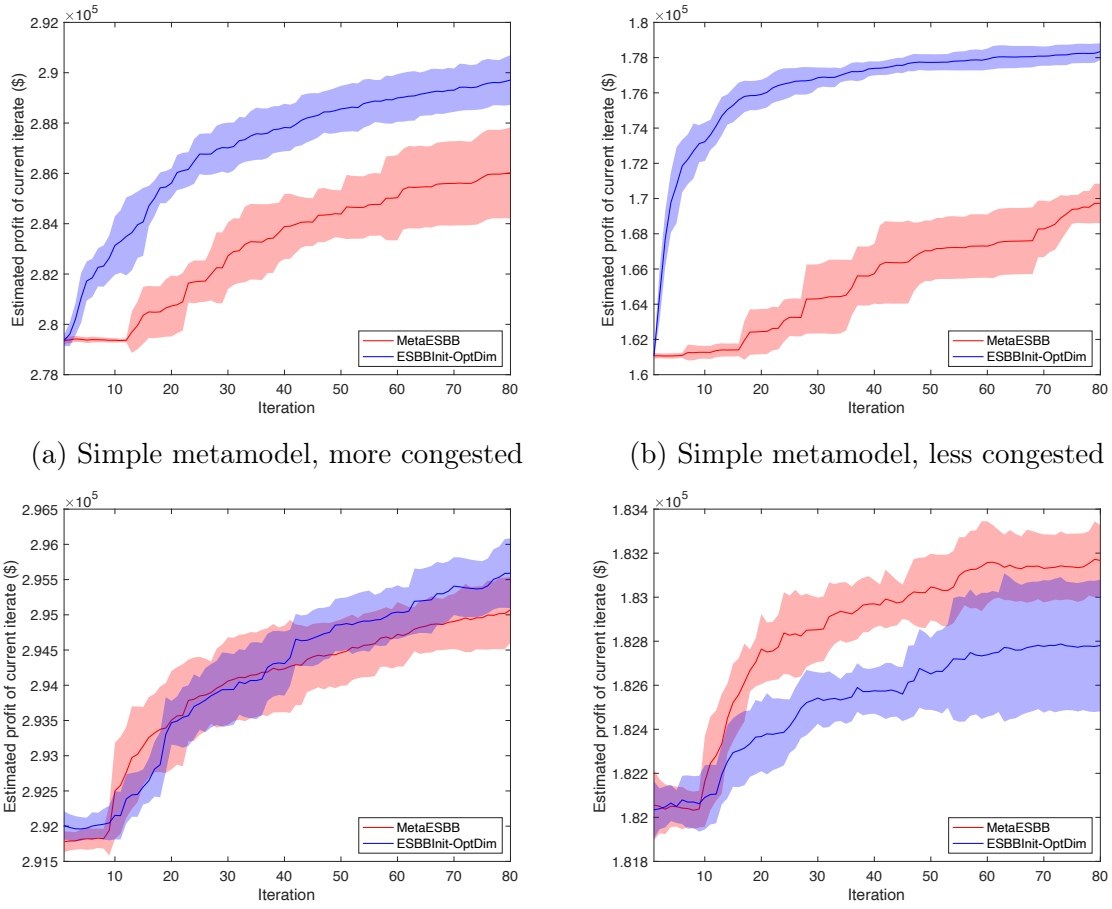
is more useful to enhance the computational efficiency of ESBB. Based on the 144 station network of Zipcar, we use four examples: (1) simple metamodel with the case of 3-time cost and 3-time demand, (2) simple metamodel with 1-time cost and 1-time demand, (3) detailed metamodel with the case of 3-time cost 3-time demand, and (4) detailed metamodel with 1-time cost and 1-time demand. Example 1 and 3 refer to a more congested case, and example 2 and 4 refer to a less congested case.

For each example, we run both algorithm 6 times. For MetaESBB, we set K = 80, v_R = 22, and v_O = 4. For ESBBInit-OptDim, we set K = 80, v_R = 14, v_L = 10 and v_O = 4. In Table 3.3, for each example, we compute the Pearson correlation coefficients between analytical and mean simulated objective values of all simulated solutions in all 12 algorithm runs (i.e., 6 runs of ESBBInit-OptDim and 6 runs of MetaESBB). We plot the comparison between the two algorithms: MetaESBB and ESBBInit-OptDim, which we initialize ESBB with the solution of the unfitted metamodel and solve the dimension-selection problem in each iteration since the second one. The results are plotted in Figure 3-15, where each subfigure represent one example. For each subfigure, the x-axis is the iteration index, and the y-axis is the estimated objective value of the current iterate. The red line is the mean performance averaged over 6 runs of MetaESBB. The red shaded area is the 95% confidence interval. The blue line is the mean performance averaged over 6 runs of ESBBInit-OptDim. The red shaded area is the 95% confidence interval.

Table 3.3: Pearson correlation coefficients between analytical and mean simulated objective values of all simulated solutions in each example

Example	Pearson Correlation Coefficient
Simple metamodel, more congested	0.8702
Simple metamodel, less congested	0.8906
Detailed metamodel, more congested	0.9851
Detailed metamodel, less congested	0.9937

We can see that in Figure 3-15a and 3-15b , the blue shaded area is higher than the red ones for both more congested and less congested cases when we use the simple metamodel. The Pearson correlation coefficients of all simulated solutions in algorithm runs of the two examples are relatively low: 0.8702 and 0.8906, respectively,



(a) Simple metamodel, more congested

(b) Simple metamodel, less congested

(c) Detailed metamodel, more congested

(d) Detailed metamodel, less congested

Figure 3-15: MetaESBB vs. ESBBInit-OptDim: use metamodel with different level of accuracy

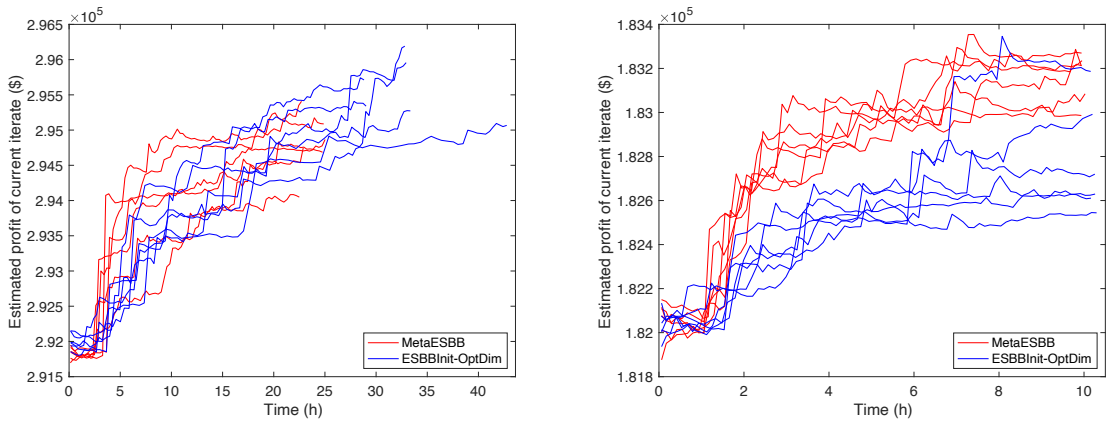
compared to the coefficients of the more detailed metamodel. This indicates that the simple model is a worse approximation to the simulator. On the other hand, when we use the detailed metamodel, in Figure 3-15c, the blue and red shaded areas overlap a lot. The Pearson correlation coefficients of all simulated solutions in algorithm runs of this example is 0.9851, the second highest among the four examples. This indicates that in this example the metamodel is more accurate. In Figure 3-15d, the red shaded area is higher than the blue shaded area, with a slight overlap at later stages. The Pearson correlation coefficients of all simulated solutions in algorithm runs of this example is 0.9937, the highest among the four examples. The Pearson correlation coefficients in Table 3.3 and the plots in Figure 3-15 indicate: (i) for metamodels with low correlation it is best to use them to identify subregions rather

than for metamodel optimization; (ii) as the metamodel correlation increases, so does the added value of using them for metamodel optimization.

As for the time issue, for the examples with simple metamodel, the time for solving the metamodel for solution and the time for solving for dimension-selection problem are both negligible, compare to the time for simulation. When using detailed metamodel for the congested case, the time for solving for solution and the time for solving for dimension-selection problem cannot be ignored. We plot the result in Figure 3-16a. The x-axis is the time, and the y-axis is the estimated objective of the current iterate. The red lines are for MetaESBB, and the blue lines are for ESBBInit-OptDim. For MetaESBB, the mean simulation time is 859.46 seconds per iteration, the mean time for solving metamodel for solutions is 95.2 seconds per iteration. For ESBBInit-OptDim, the mean simulation time is 888.3 seconds per iteration, the mean time for solving dimension-selection problem is 599.1 seconds per iteration. When using detailed metamodel for less congested case, we plot the results in Figure 3-16b. The figure format is the same as Figure 3-16a. In this example, the time for solving the metamodel for solution and the time for solving for dimension-selection problem are both small, compared to simulation time. For MetaESBB, the mean simulation time is 422.4 seconds per iteration, the mean time for solving metamodel for solutions is 10.4 seconds per iteration. For ESBBInit-OptDim, the mean simulation time is 422.5 seconds per iteration, the mean time for solving dimension-selection problem is 25.6 seconds per iteration.

3.3.3 High-dimensional case study with 315 Stations

In this section, we study the performance of the proposed algorithm using a case with 315 stations, which are the same stations we used in Section 2.3.3 and 2.3.4. The location of these stations are presented in Figure 2-8. We set the demand level to 3 times of the trained demand for each station and the cost level to 3 times of the real cost level. The fleet size is 894 (i.e., $X = 894$). We compare the performance of MetaESBB-OptDim, MetaESBB and MetaESBB-RanDim. For MetaESBB-OptDim and MetaESBB-RanDim, we set $v_R = 18$, $v_L = 20$, $v_O = 5$ and $U = 5$. For MetaESBB,



(a) Detailed metamodel, more congested case (b) Detailed metamodel, less congested case

Figure 3-16: MetaESBB vs. ESBBInit-OptDim: compare algorithms using elapsed time

we set $v_R = 28$ and $v_O = 5$. For all algorithm runs, we set $K = 40$, $\Delta n_F = 5$ and $\Delta n_A = 2$.

In Figure 3-17, we compare the performance of the three algorithms using the simple metamodel. We run each algorithm 6 times. The red line shows the mean performance of MetaESBB-OptDim, with the red shaded area as the 95% confidence interval. The blue (black) line and blue (gray) shaded area have similar meanings, but correspond to MetaESBB-RanDim (MetaESBB).

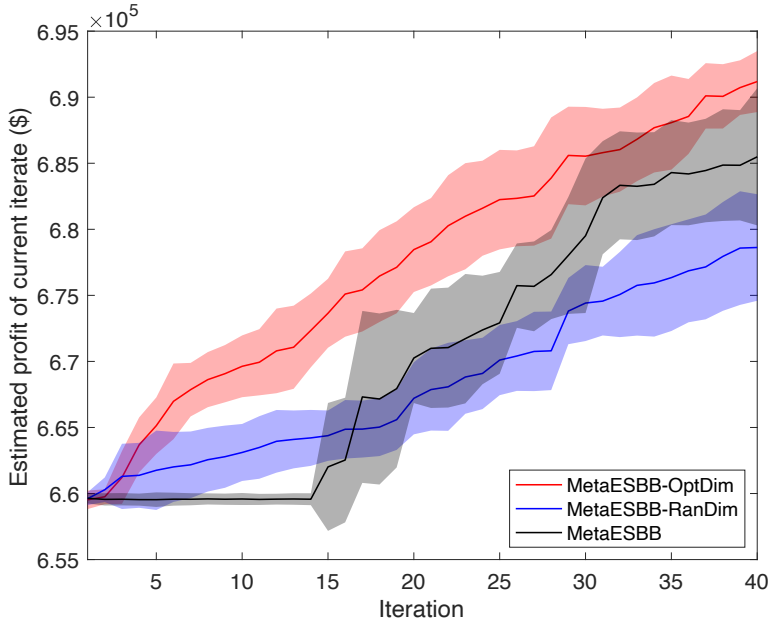


Figure 3-17: MetaESBB-OptDim vs. MetaESBB-RanDim vs. MetaESBB (simple metamodel)

In Figure 3-17, we see that the red shaded area are above the black shaded area for most iterations. The black shaded shows overlap with the red shaded area at latter iterations, but the black line is under the red line. This shows that MetaESBB-OptDim outperforms MetaESBB. For MetaESBB-OptDim, about 9% of the current iterate are obtained by solving the current iterate, and 91% are identified by sampling from the lower-dimensional subregion. All current iterate of MetaESBB are obtained by solving the fitted metamodel. These findings are consistent with the findings in Section 3.3.2 This shows that when the metamodel is less accurate, sampling from the lower-dimensional subregion built by solving the dimension-selection problem is more useful than solving the fitted metamodel.

In Figure 3-17, we also see that the blue shade area are lower than the red shaded area throughout the algorithm runs. This shows that MetaESBB-OptDim outperforms MetaESBB-RanDim For MetaESBB-RanDim, 13% of the current iterate are identified by solving the fitted metamodel, while 87% are found by sampling from the lower-dimensional subregion. This indicates that sampling from a lower-subregion built by randomly select dimensions may be less efficient than sampling from a lower-subregion built by solving the dimension-selection problem. The gray shaded area also becomes higher than the blue shaded area at latter stages. This shows that MetaESBB outperforms MetaESBB-RanDim. This indicates that sampling from lower-subregion built by randomly select dimensions may have negative effect on the quality of metamodel such that solving the fitted metamodel becomes less useful to find better solutions.

Using the detailed metamodel, we run MetaESBB-OptDim, MetaESBB and MetaESBB-RanDim each for 3 times. In Figure 3-18, we compare MetaESBB-OptDim (red lines) and MetaESBB (black lines). The x-axis is the iteration index, and the y-axis the the simulated mean objective value of the current iterate. At earlier stage, the red lines are higher, but then the black lines catch up and become higher than the red lines. The red lines outperforms the black lines at latter stages of the algorithm run. For MetaESBB-OptDim, 74% of the current iterates are identified by solving the fitted metamodel, while 26% of them are found by sampling from the lower-dimensional

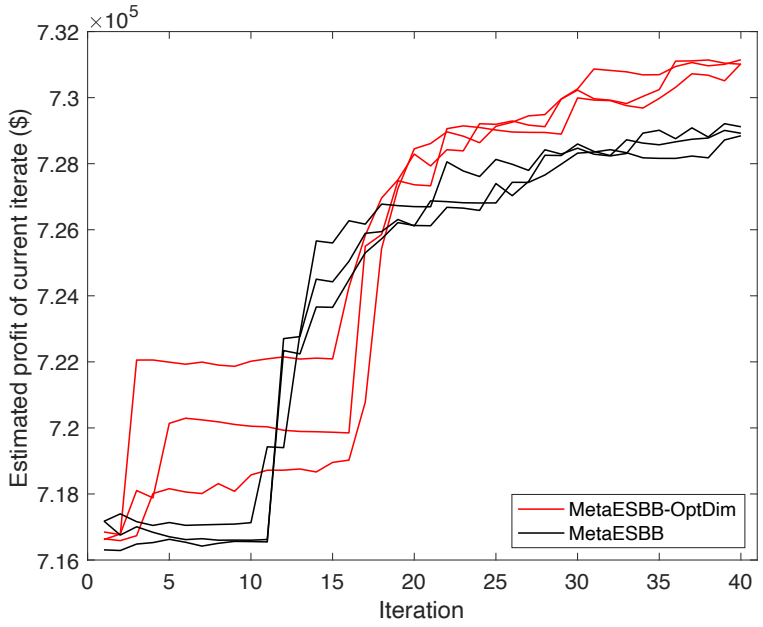


Figure 3-18: MetaESBB-OptDim vs. MetaESBB (detailed metamodel)

subregion. For MetaESBB, all current iterates are obtained by solving the fitted metamodel. This means that given the same simulation budget, using metamodel to build the lower-dimensional subregion and sampling from it can improve the performance of MetaESBB, when we use a more accurate metamodel. This is consistent with the findings in Section 3.3.2.

In Figure 3-19, we compare MetaESBB-OptDim (red lines) and MetaESBB-RanDim (blue lines). The axis format is the same as the axis format of Figure 3-18. We find that the red lines outperform the blue lines, but their performance are close. For MetaESBB-RanDim, 62% of current iterates are found by solving the fitted metamodel, and 38% of them are identified by sampling from the lower-dimensional subregion. This means that given the same simulation budget, MetaESBB-OptDim is slightly better than MetaESBB-RanDim. This is also consistent with the findings in Section 3.3.2.

In this case of 315 stations, the time for solving the dimension-selection problem using the detailed metamodel is no longer negligible compared to the time for simulation. In Figure 3-20, we plot the algorithm runs of MetaESBB-OptDim (red lines), MetaESBB-RanDim (blue lines) and MetaESBB (black lines) The x-axis is the time.

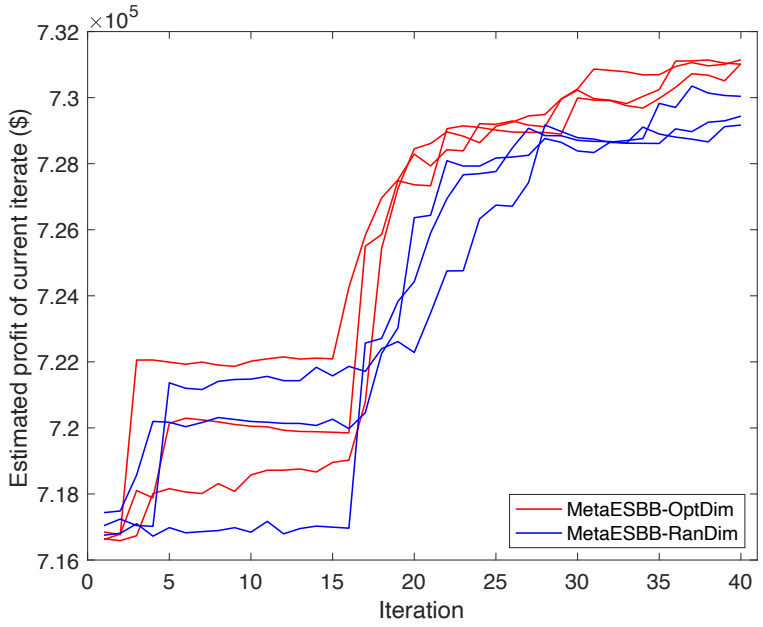


Figure 3-19: MetaESBB-OptDim vs. MetaESBB-RanDim (detailed metamodel)

The y-axis the simulated mean objective value of the current iterate. We can see that several red lines are longer. For the 3 runs of MetaESBB-OptDim using the detailed metamodel, the time for solving one dimension-selection problem is 1144.1 seconds on average. The simulation time per iteration on average is 2987.4 seconds. Consider the time cost of solving the dimension-selection problem, using lower-dimensional subregion built by randomly select dimensions may be a better choice to enhance the performance of MetaESBB, when we are using a more accurate metamodel.

3.4 Summary

This chapter proposes MetaESBB-OptDim, which extends ESBB, a globally convergent discrete SO algorithm, to address the problem of car-sharing network design. It is a metamodel SO algorithm, which combines a metamodel with problem specific information with a general-purpose globally convergent discrete SO algorithm ESBB. We use the metamodel for two purposes: (1) solving the updated metamodel in each iteration for solutions and (2) select dimensions to build a lower-dimensional subregion for sampling. We use two metamodels, one simple and one detailed, with

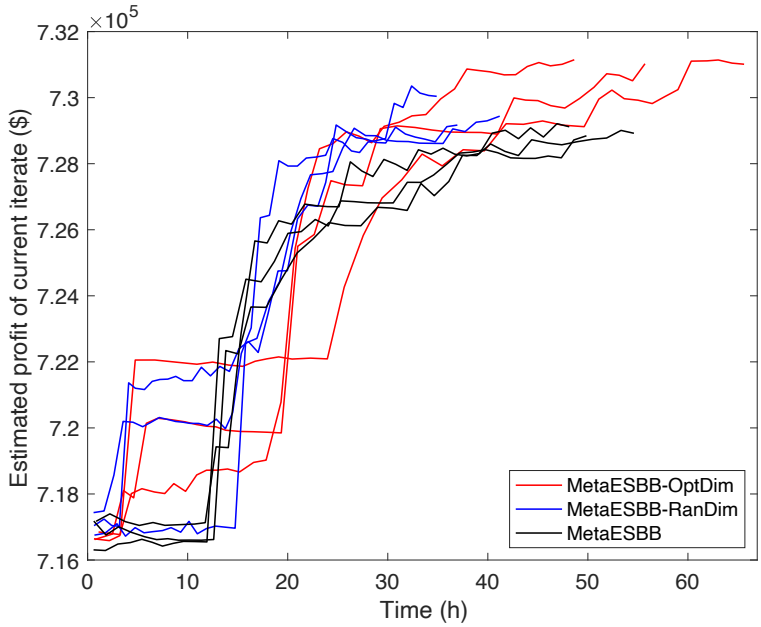


Figure 3-20: Change of current iterate objective value over time: MetaESBB-OptDim vs. MetaESBB-RanDim vs. MetaESBB (detailed metamodel)

different ability to approximate the simulator. And we use toy networks to show the accuracy of the two metamodels by comparing the analytical objective values and mean simulated objective values of all feasible solutions.

We apply the proposed algorithm to high-dimensional case studies using Zipcar's historical data of its Boston fleet. Experiment results indicate that metamodel can bring a good initial start point to ESBB. In the case study, the solution of both unfitted simple metamodel and detailed metamodel are proved to be better than randomly sampled solutions. Chapter 2 shows the value of solving fitted metamodel in each iteration. In this works, we show that use the metamodel to select dimensions to build lower-dimensional sampling subregion can further enhance the algorithm computational efficiency. If the time for solving the dimension-selection problem is acceptable, we recommend to do so. We also show that using the metamodel to build lower-dimensional subregions for sampling can also improve the efficiency of ESBB. It has better performance than using lower-dimensional subregion built by randomly select dimensions, given the same computational budget. We find that when metamodel is more accurate, using it to solve for solutions is more likely to find

better solutions than using it to build a lower-dimensional subregion for sampling, but otherwise when the metamodel is less accurate.

Chapter 4

Conclusions

This work proposes discrete SO algorithms for a family of large-scale car-sharing network design problems. We sample from disaggregate reservation data, and tackle the corresponding service design optimization problems directly using the disaggregate data, without aggregating it. We propose two approaches that rely on the general idea of metamodeling. We show that the use of metamodels allows the discrete SO algorithms to become scalable and computationally efficient.

In Chapter 2, we propose a metamodel which has the form of a MIP to approximate the simulated objective function. This metamodel has a physical part with analytically compute the profit of the car-sharing system, and functional part which is a linear expression of decision variables. We propose MetaAHA, which embeds the metamodel into AHA - an efficient locally convergent general-purpose discrete SO algorithm. In each iteration, we solve the metamodel to obtain new solutions and update the metamodel parameters using simulation results. Experiments with a high-dimensional case studies show that the metamodel can identify points with good performs both initially (i.e., as initial points) as well as throughout the iterations of the discrete SO algorithm. We also benchmark MetaAHA with stochastic programming (SP).

In Chapter 3, we propose MetaESBB-OptDim, which embeds the metamodels into ESBB, a globally convergent discrete SO algorithm. We maintain the globally convergent property of ESBB. We use the metamodel for two purposes: solving the

updated metamodel in each iteration for solutions, and solving a metamodel-based dimension-selection problem to build a lower-dimensional subregion and sample from it. We used two metamodels with different accuracy of approximating the simulated objective value. Experiments with high-dimensional problems show an added value of using the lower-dimensional subregion, given that we solve the metamodel for solutions in each iteration. Sampling from the lower-dimensional subregion is more efficient than solving the fitted metamodel for solution when the metamodel is less accurate. But when the metamodel is more accurate, sampling from the lower-dimensional subregion is less efficient than solving the fitted metamodel for solution.

We view this general idea of combining analytical MIP formulations with general-purpose SO algorithms, or more broadly with general-purpose sampling strategies, as an innovative and promising area of future research. With the increase in the availability and the resolution of transportation data comes the potential to address more intricate formulations of traditional transportation optimization problems (e.g., formulations with a more detailed probabilistic data-driven description of demand). This work illustrates how the traditional MIP formulations that exist can be coupled with high-resolution data, a sampling (or simulation) strategy, and a general-purpose SO algorithm, to address this next generation of transportation problems.

There is a wide-variety of general-purpose discrete SO algorithms. As general-purpose algorithms, they can be used to address a broad class of problems. Nonetheless, they are rarely designed such as to achieve good short term performance (i.e., good performance within few simulation runs). This dissertation illustrates how the scalability, computational efficiency and robustness of these SO algorithms can be enhanced such as to enable them to address realistic transportation problems at scale.

The proposed approach performs optimization preserving the disaggregate information in the data (rather than limiting its use to fitting aggregate demand parameters). This leads to a data-driven approach that exploits the rich information of demand and of demand-supply interactions embedded in the data. Nonetheless, this also limits its use for car-sharing markets where data is unavailable or unreliable. In particular, it is not directly applicable for new markets where data has not yet been

collected.

The propose algorithms can also be used in other fields beyond transportation (e.g., Wang et al. (2019)). Extensions of ongoing interest include the use of MIPs to enable the design of real-time discrete SO problems. For future work, it is worth to further study how many dimensions we should choose to build the lower-dimensional subregion for discrete SO problems with different dimensionality. In our method, the lower-dimensional subregion are build around the current iterate, hence it searches around the current iterate locally. It is also worthwhile to study other possible local search method that are suitable to combine with metamodel-based globally convergent discrete SO algorithm.

Appendix A

Metamodel fitting

We use a method similar to the one used by Osorio and Bierlaire (2013) to compute the value of metamodel parameter $\boldsymbol{\beta}$ in each iteration. Specifically, at the Step 4b of MetaAHA (i.e., Algorithm 1) and the Step 5c of both MetaESBB (i.e., Algorithm **??**) and MetaESBB-OptDim (i.e., Algorithm 2), we solve the following optimization problem

$$\begin{aligned}\min_{\boldsymbol{\beta}_k} \quad & \sum_{\boldsymbol{x}\in\mathcal{G}^k}\left\{w_k(\boldsymbol{x})\left[\hat{g}(\boldsymbol{x};\boldsymbol{q}_1)-\left(\beta_{k,0}g_A^*(\boldsymbol{x};\boldsymbol{q}_2)+\beta_{k,1}+\sum_{i\in\mathcal{I}}\beta_{k,i+1}x_i\right)\right]\right\}^2 \\ & +\left(w_0(\beta_{k,0}-1)\right)^2+\sum_{i=1}^{|\mathcal{I}|+1}(w_0\beta_{k,i})^2.\end{aligned} \tag{A.1}$$

$\mathcal{G}^k$ is the set containing all solutions that have been simulated up until iteration k. w_0 is a fixed parameter. $w_k(\boldsymbol{x}) = (1+\|\boldsymbol{x}-\boldsymbol{x}_k\|)^{-1}$ is a weight parameter reflecting the Euclidean distance between a solution $\boldsymbol{x}$ and the current iterate $\boldsymbol{x}_k$. Hence, solutions closer to the current iterate are considered more important than solutions that are more distant. $\hat{g}(\boldsymbol{x};\boldsymbol{q}_1)$ is the mean simulated objective value of solution $\boldsymbol{x}$. $g_A^*(\boldsymbol{x};\boldsymbol{q}_2)$ is the analytical objective value of $\boldsymbol{x}$, which is obtained by maximizing $g_A(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2)$ over $\boldsymbol{z}$ with constraint $h(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_2)=0$ (i.e., Equation (3.6)) and $\boldsymbol{x}$ fixed.

This least square problem minimizes the weighted distance between the simulated objective values and the analytical objective values of all solutions that have been simulated so far. The last two terms will make sure that the least-square matrix is

full rank. For the term $(w_0(\beta_{k,0} - 1))^2$, the intuition is to assume that the analytical model is a good approximation to the mean simulated objective value for the feasible solutions. We also have the intercept term $\beta_{k,1}$. The reason to have this intercept term is to better capture the systematic bias of the analytical approximation. We also note that this may result in a metamodel with objective value different to the simulated objective value. For example, when the decision vector $\boldsymbol{x} = \mathbf{0}$, the simulated objective value will be 0, but the metamodel objective value will equal to $\beta_{k,1}$ rather than 0. This problem is a continuous convex optimization problem, and can be efficiently solved.

Appendix B

MetaAHA algorithmic details

In this section, we present algorithmic details of MetaAHA. The algorithmic steps refer to Algorithm 1. In Step 2 of the algorithm, the number of simulation replications to run for a given point $\boldsymbol{x}$ up until and including iteration k, denoted $N_k(\boldsymbol{x})$, is computed based on the approach of AHA (Xu et al., 2013). It is given by $N_k(\boldsymbol{x}) = \min\{5, \lceil 5(\log k)^{1.01} \rceil\}$. If at a given iteration k, the number of simulation replications of point $\boldsymbol{x}$ obtained from previous iterations is greater or equal to $N_k(\boldsymbol{x})$, then we do not evaluate additional replications.

In Step 2 of the algorithm, the hyperbox is updated based on the following AHA approach (Xu et al., 2013). Let $\boldsymbol{x}_k$ denote the current iterate at iteration k, with the ith element denoted $x_{k,i}$. Let $\mathcal{E}(k)$ denote the set of points that have been simulated up until and including iteration k. The hyperbox is defined (or updated) at iteration k as $\mathcal{H}_k = \{\boldsymbol{x} : l_{k,i} \leq x_i \leq u_{k,i}, \forall i \in \mathcal{I}\}$. The bounds $l_{k,i}$ and $u_{k,i}$ are defined as follows.

$$l_{k,i} = \max_{\boldsymbol{x} \in \mathcal{E}(k) \setminus \{\boldsymbol{x}_k\}} \{x_i : x_i < x_{k,i}\}, \forall i \in \mathcal{I}.$$

If $l_{k,i}$ is empty, then set $l_{k,i} = 0$. Similarly,

$$u_{k,i} = \min_{\boldsymbol{x} \in \mathcal{E}(k) \setminus \{\boldsymbol{x}_k\}} \{x_i : x_i > x_{k,i}\}, \forall i \in \mathcal{I}.$$

If $u_{k,i}$ is empty, then set $u_{k,i} = N^i$.

Step 4b of the algorithm fits the metamodel parameter $\boldsymbol{\beta}_k$ by solving the least squares problem discussed in Appendix A.

Figure B-1 provides a flowchart summary of the MetaAHA algorithm.

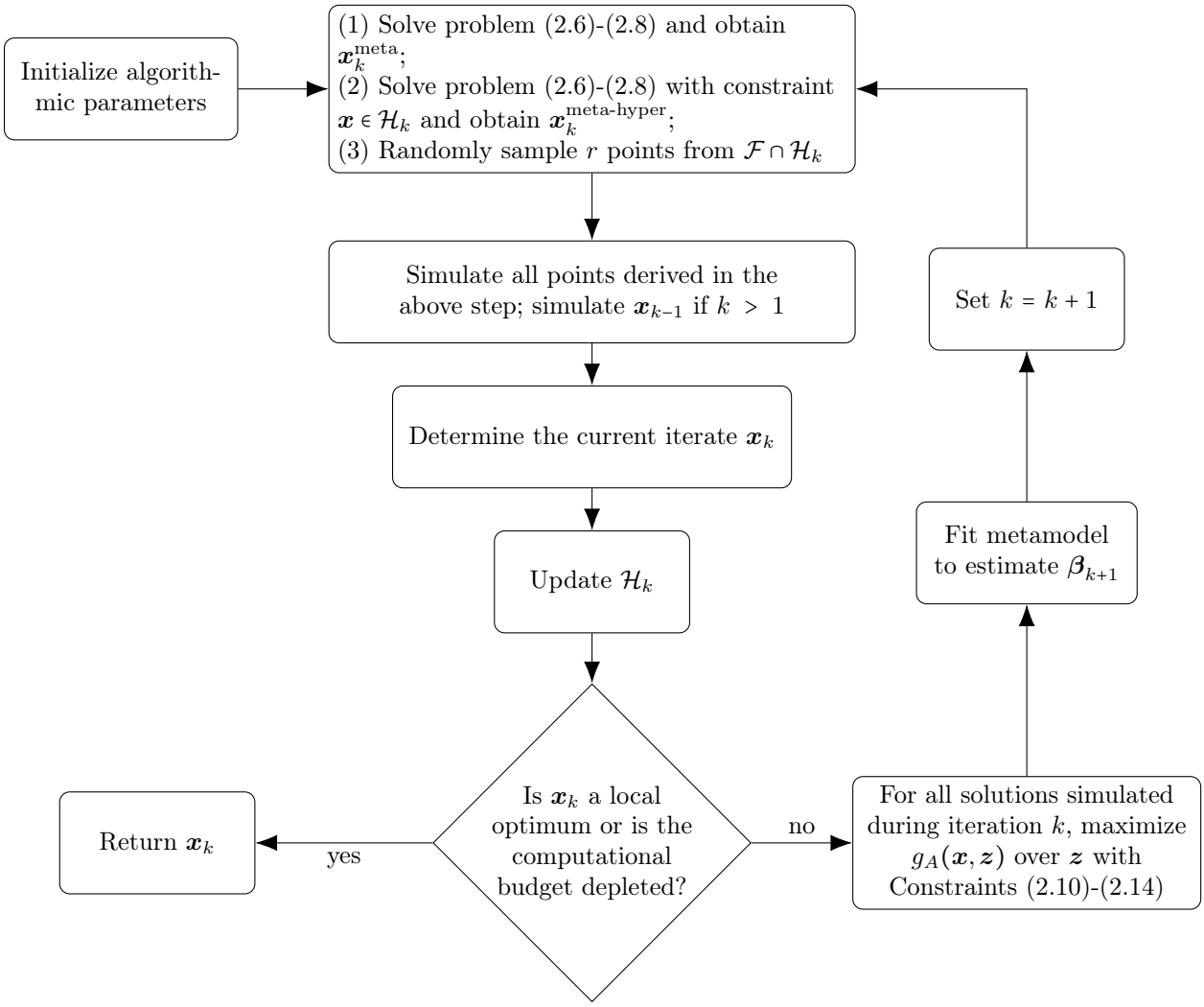


Figure B-1: MetaAHA Steps

Appendix C

Stochastic programming (SP) formulation

To formulate the SP model, we use the notation of the main manuscript and introduce the following notation.

- Q: number of demand scenarios;
- $d_{tl}^{i(q)}$: number of customers that desire a reservation at station i with start time t and duration l in demand scenario q;
- $z_{tl}^{i(q)}$: number of customers that make a reservation at station i with start time t and duration l in demand scenario q;
- $z_{tl}^{ji(q)}$: number of customers that desire to make a reservation at station j with start time t and duration l but make an adjusted reservation at station i with start time t and duration l in demand scenario q;
- $\boldsymbol{z}$: vector that combines all variables $\{z_{tl}^{i(q)}\}$ and $\{z_{tl}^{ji(q)}\}$;
- $\pi^{(q)}$: probability of scenario q, set to $1/Q$;
- $\boldsymbol{q}_3$: vector of exogenous parameters;
- g_{SP}: analytical approximation of g (Equation (2.2)) derived by the SP model.

We view the network design strategy $\boldsymbol{x}$ as the first-stage decision variables, and the demand-supply interaction $\boldsymbol{z}$ as the second-stage decision variables. The SP problem is formulated as follows.

$$\max_{\boldsymbol{x},\boldsymbol{z}} \quad g_{SP}(\boldsymbol{x},\boldsymbol{z};\boldsymbol{q}_3) = \sum_{q=1}^{Q} \pi^{(q)} \left(\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}_i} \sum_{t\in\mathcal{T}} \sum_{l\in\mathcal{L}} p^{ij} r_{tl} z_{tl}^{ij(q)} \right) - \sum_{i\in\mathcal{I}} c_i x_i, \tag{C.1}$$

subject to

$$\sum_{j\in\mathcal{I}_i} z_{tl}^{ji(q)} = z_{tl}^{i(q)} \qquad \forall i\in\mathcal{I}, \forall t\in\mathcal{T}, \forall l\in\mathcal{L}, \forall q\in\{1,2,\ldots,Q\}, \tag{C.2}$$

$$\sum_{j\in\mathcal{I}_i} z_{tl}^{ij(q)} \le d_{tl}^{i(q)} \qquad \forall i\in\mathcal{I}, \forall t\in\mathcal{T}, \forall l\in\mathcal{L}, \forall q\in\{1,2,\ldots,Q\}, \tag{C.3}$$

$$\sum_{l\in\mathcal{L}} z_{tl}^{i(q)} + \sum_{l\in\mathcal{L}} \sum_{t'\in\mathcal{T}_1(t,l)} z_{t'l}^{i(q)} \le x_i \qquad \forall i\in\mathcal{I}, \forall t\in\mathcal{T}, \forall q\in\{1,2,\ldots,Q\}, \tag{C.4}$$

$$z_{tl}^{i(q)} \in \mathbb{R}_+ \qquad \forall i\in\mathcal{I}, \forall t\in\mathcal{T}, \forall l\in\mathcal{L}, \forall q\in\{1,2,\ldots,Q\}, \tag{C.5}$$

$$z_{tl}^{ij(q)} \in \mathbb{R}_+ \qquad \forall i\in\mathcal{I}, \forall j\in\mathcal{I}_i, \forall t\in\mathcal{T}, \forall l\in\mathcal{L}, \forall q\in\{1,2,\ldots,Q\}, \tag{C.6}$$

$$\boldsymbol{x}\in\mathcal{F}, \tag{C.7}$$

where $\mathcal{T}_1(t,l) = \{t'\in\mathcal{T} : t'+1 \le t \le t'+l-1\}$. In this model, the exogenous parameters are $d_{tl}^{i(q)}$ and $\pi^{(q)}$, as well as r_{tl}, c_i, p^{ij}, $t_{\max}$ and $l_{\max}$ defined in Section 2.2.3, represented by the vector $\boldsymbol{q}_3$. The objective function (C.1) is the expected revenue over all scenarios minus the cost. Constraints (C.2)-(C.6) are the equivalent of their MIP counterparts Constraints (2.10)-(2.14), respectively. Constraint (C.7) is equivalent to Constraint (2.8).

Appendix D

Sampling of feasible solutions for the experiments of Figures 2-12 and 2-13

For a given demand scenario, let $\boldsymbol{x}^{(1)}$ be the SP solution and $\boldsymbol{x}^{(2)}$ be the best MetaAHA solution of the experiments of Section 2.3.4. We use the following Algorithm 3 to generate a solution near the line connecting the SP optimal solution and the best MetaAHA solution:

Algorithm 3 Sampling one integral point close to the line connecting two points

- Step 1: generate $u \sim U(0,1)$, where $U(0,1)$ is the standard uniform distribution.
- Step 2: let $\tilde{\boldsymbol{x}} = \boldsymbol{x}^{(1)} + u\left(\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}\right)$.
- Step 3: build a hyperbox $\mathcal{H}(\tilde{\boldsymbol{x}}) = \{\boldsymbol{x} : \tilde{x}_i - 2 \leq x_i \leq \tilde{x}_i + 2, \forall i \in \mathcal{I}\}$.
- Step 4: randomly sample a point $\boldsymbol{x}$ from $\mathcal{H}(\tilde{\boldsymbol{x}}) \cap \mathcal{F}$ using the uniform sampling distribution of AHA (Xu et al., 2013).

Appendix E

A generic partition method

We present a generic partition method for the ESBB algorithm. We have two control parameters: (1) the number of total dimensions to select, which we denote as D; (2) the proportion of area in the newly generated subregion which contains the current iterate, which denote as κ ($\kappa \in [0, 1]$). A simple 2-D demonstration of this partition method are shown in Figure E-1. The black dot represent the current iterate. In this case, we have $D = 2$ and $\kappa = 0.5$. We first select the horizontal dimension, cut the feasible region into two subregions in a way that the current iterate is in the interior part of one of the subregions. Since $\kappa = 0.5$, the cut is in the center between the current iterate and the left edge of the feasible region along the horizontal axis. Then we select the vertical dimension and cut the subregion that contains the current iterate in a similar way. We finally get 3 subregions, and the current iterate is in the relative interior part of one of them.

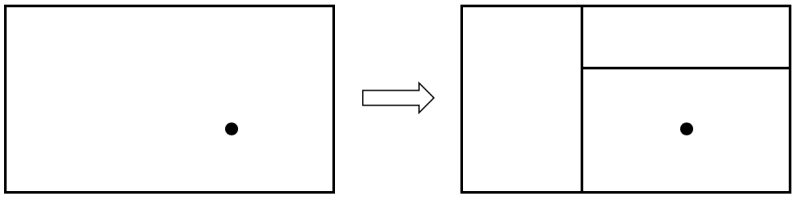


Figure E-1: 2D illustration of the partition method

Here we formally define the generic parallel partitioning algorithm we mentioned in . Let $\tilde{\boldsymbol{x}}$ be the current iterate. Let $\mathcal{R} = \mathcal{F} \cap \mathcal{H}$ be the current best subregion,

where $\mathcal{H} = \{\boldsymbol{x} \in \mathbb{R}^{|\mathcal{I}|} : LB_i \leq x_i \leq UB_i\}$ is a hyperbox that bounds the decision vector along each dimension. The generic parallel partitioning algorithm is presented in Algorithm 4.

Algorithm 4 A generic partitioning algorithm

Initialization:

- Let $\mathcal{R}^* = \mathcal{R}$, $\mathcal{D} = \{i \in \mathcal{I} : LB_i < UB_i\}$, and $\mathcal{P}(\mathcal{R}) = \{\mathcal{R}^*\}$. Let $k = \min(D, |\mathcal{D}|)$.

Step 1: select dimension

- Randomly select $i \in \mathcal{D}$, let $l = \tilde{x}_i - LB_i$, $u = UB_i - \tilde{x}_i$, $\mathcal{D} = \mathcal{D} \setminus \{i\}$ and $\mathcal{P}(\mathcal{R}) = \mathcal{P}(\mathcal{R}) \setminus \{\mathcal{R}^*\}$.

Step 2: partitioning

- If $l > u$, let $\mathcal{R}^u = \mathcal{R}^* \cap \{\boldsymbol{x} \in \mathcal{H} : x_i \geq \min\left(UB_i, \lfloor \tilde{x}_i - (\tilde{x}_i - LB_i)\kappa \rfloor + 1\right)\}$, $\mathcal{R}^l = \mathcal{R}^* \cap \{\boldsymbol{x} \in \mathcal{H} : x_i \leq \lfloor \tilde{x}_i - (\tilde{x}_i - LB_i)\kappa \rfloor\}$ and $\mathcal{R}^* = \mathcal{R}^u$.
- If $l < u$, let $\mathcal{R}^u = \mathcal{R}^* \cap \{\boldsymbol{x} \in \mathcal{H} : x_i \geq \lceil \tilde{x}_i + (UB_i - \tilde{x}_i)\kappa \rceil\}$, $\mathcal{R}^l = \mathcal{R}^* \cap \{\boldsymbol{x} \in \mathcal{H} : x_i \leq \max\left(LB_i, \lceil \tilde{x}_i + (UB_i - \tilde{x}_i)\kappa \rceil - 1\right)\}$ and $\mathcal{R}^* = \mathcal{R}^l$.
- If $l = u$, let s be a randomly and uniformly sampled number from $[0, 1]$. If $s > 0.5$, follow the rule of the case $l > u$; otherwise, follow the rule of $l < u$.

Step 3: update partition and update iteration counter

- Step 3a: Let $\mathcal{P}(\mathcal{R}) = \mathcal{P}(\mathcal{R}) \cup \{\mathcal{R}^l, \mathcal{R}^u\}$.
- Step 3b: Set $k = k - 1$. If $k = 0$, terminate; otherwise, proceed to Step 1.

To initialize the algorithm, we identify the dimensions along which we can do further partition for the current best subregion $\mathcal{R}$. Then we randomly pick one of them to partition. On the selected dimension, we cut $\mathcal{R}$ along this dimension and try to have the current iterate in the interior part of one of the generated subregions. Not that the current iterate may still be on the boarder of the newly generated subregion. Then we add both newly generated partitions to $\mathcal{P}(\mathcal{R})$, the set of all subregions that will be generated for $\mathcal{R}$ for running this algorithm. Then we further partition the subregion that contains the current iterate.

We design the partition method in this way because we want to keep the current

iterate in the relative interior part of the current best subregion. Also, we can use the parameters D and κ to control the speed of convergence. Note that this method can ensure that the generated subregions have no common elements, and the union of these subregions will be $\mathcal{R}$. In all experiments in Section 3.3.2 and 3.3.3, we set $D = 5$ and $\kappa = 0.5$.

Appendix F

RanDim: build lower-dimensional subregion by randomly select dimensions

In order to show the effectiveness of using the metamodel to selection dimensions optimally, we benchmark it with a method that build the lower-dimensional subregion by randomly select dimensions. Specifically, in iteration k ($k > 1$), given $\mathcal{R}^{k-1}$, the best subregion obtained at the end of iteration $k-1$, we first find $\mathcal{D}^k = \{i \in \mathcal{I} : LB_i^{k-1} < UB_i^{k-1}\}$, and then randomly select $\min(U, |\mathcal{D}^k|)$ unique elements from $\mathcal{D}^k$ to form $\mathcal{D}_*^k$. Let $\bar{\boldsymbol{u}}_k$ be a vector such that $\bar{u}_{ik} = 1$ if $i \in \mathcal{D}_*^k$ and $\bar{u}_{ik} = 0$ if $i \notin \mathcal{D}_*^k$. We create the random hyperbox of iteration k as

$$\mathcal{H}_k^{\text{ran}} = \left\{\boldsymbol{x} : LB_i^{k-1}\bar{u}_{ik} + x_i^{k-1}(1-\bar{u}_{ik}) \le x_i \le UB_i^{k-1}\bar{u}_{ik} + x_i^{k-1}(1-\bar{u}_{ik}), \forall i \in \mathcal{I}\right\}. \quad \text{(F.1)}$$

The main difference between $\mathcal{H}_k^{\text{opt}}$ and $\mathcal{H}_k^{\text{ran}}$ is that the former one is obtained by solving a combinatorial optimization problem, while the latter one is generated randomly.

We propose an algorithm MetaESBB-RanDim. It is almost same to MetaESBB-OptDim (i.e., Algorithm 2). The only difference between MetaESBB-RanDim and MetaESBB-OptDim is that at Step 2e, MetaESBB-RanDim randomly generates a

hyperbox $\mathcal{H}_k^{\text{ran}}$ and then randomly samples solutions from $\mathcal{H}_k^{\text{ran}} \cap \mathcal{R}^{k-1}$. If we remove Step 2e from MetaESBB-RanDim, it is the same as MetaESBB.

Appendix G

Detailed solution comparison of Section 3.3.2

To further investigate the performance of MetaESBB-OptDim, MetaESBB and MetaESBB-RanDim, we run additional simulation replications for solutions we obtain in Section 3.3.2. In this appendix, we conduct statistical test to compare the performance of those solutions.

In Section 3.3.2, we obtain 6 MetaESBB-OptDim solutions, 6 MetaESBB solutions, and 6 MetaESBB-Random solutions using the simple metamodel. For each of the 18 solutions, we run 50 simulation replications, and compute the mean simulated profit and sample standard deviation based on the 50 replication. The results are summarized in Table G.1.

In Table G.2, we compare all 6 MetaESBB-OptDim final solutions to all 6 MetaESBB solutions when using the simple metamodel. Each row of the table corresponds to a solution of MetaESBB-OptDim, and each column corresponds to a solution of MetaESBB. In each cell, we show the p-value of a one-way t-test using the 50 simulated objective value of the corresponding row solution and the 50 simulated objective value of the corresponding column solution. The null hypothesis is that the mean simulated objective value of the row solution is the equal to the mean simulated objective value of the column solution. The alternative hypothesis is that the former one is larger than the latter one, as we solve a maximization problem in Section 3.3.2.

Table G.1: Final solution simulation: MetaESBB-OptDim, MetaESBB-RanDim and MetaESBB (simple metamodel)

Algorithm	Run	Mean	Standard deviation
MetaESBB-OptDim	1	289422.06	346.64
	2	288960.37	375.45
	3	288197.96	354.66
	4	287885.95	433.10
	5	287987.78	341.29
	6	288172.13	350.00
MetaESBB	1	288007.14	404.47
	2	286407.97	412.11
	3	287697.99	406.99
	4	283734.09	328.65
	5	285946.58	323.18
	6	284136.31	359.28
MetaESBB-RanDim	1	288193.08	310.05
	2	288393.21	330.27
	3	288732.76	354.05
	4	287917.01	324.43
	5	287650.01	324.85
	6	287164.43	363.40

If the p-value is larger than 0.05, the value is shown in bold and we do not reject the null hypothesis, otherwise we reject the null hypothesis. We see that in Table G.2, only two values are in bold and we cannot reject the null hypothesis, while for the other 34 t-tests we can reject the null hypothesis.

Table G.2: Final solution comparison: MetaESBB-OptDim vs. MetaESBB (simple metamodel)

Run		MetaESBB					
		1	2	3	4	5	6
MetaESBB-OptDim	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	3	0.0069	0.0000	0.0000	0.0000	0.0000	0.0000
	4	**0.9243**	0.0000	0.0138	0.0000	0.0000	0.0000
	5	**0.6017**	0.0000	0.0001	0.0000	0.0000	0.0000
	6	0.0158	0.0000	0.0000	0.0000	0.0000	0.0000

In Table G.3, we compare the 6 solutions of MetaESBB-OptDim to the 6 solutions of MetaESBB-RanDim when using the simple metamodel. The table format is the same as the format of In Table G.2, except that now the column solutions are solutions

of MetaESBB-RanDim. In Table G.3, there are 14 values in bold, which means that we cannot reject the null hypothesis of the 14 t-tests. This is consistent with Figure 3-12, where the red shades area have some overlap with the blue shaded area at the last iteration.

Table G.3: Final solution comparison: MetaESBB-OptDim vs. MetaESBB-RanDim (simple metamodel)

Run		MetaESBB					
		1	2	3	4	5	6
MetaESBB-OptDim	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	2	0.0000	0.0000	0.0012	0.0000	0.0000	0.0000
	3	**0.4709**	**0.9973**	**1.0000**	0.0000	0.0000	0.0000
	4	**1.0000**	**1.0000**	**1.0000**	**0.6571**	0.0014	0.0000
	5	**0.9989**	**1.0000**	**1.0000**	**0.1453**	0.0000	0.0000
	6	**0.6239**	**0.9992**	**1.0000**	0.0001	0.0000	0.0000

In Section 3.3.2, we obtain 6 MetaESBB-OptDim solutions, 6 MetaESBB solutions, and 6 MetaESBB-Random solutions using the detailed metamodel. For each of the 18 solutions, we also run 50 simulation replications, and compute the mean simulated profit and sample standard deviation based on the 50 replication. The results are summarized in Table G.4.

In Table G.5, we compare the 6 solutions of MetaESBB-OptDim to the 6 solutions of MetaESBB when using the detailed metamodel. The format of Table G.5 is the same to the Table G.2 We can see that no value in Table G.5 are in bold. This means that we reject the null hypothesis for 36 t-tests between the solutions of MetaESBB-OptDim and MetaESBB. All 6 MetaESBB-OptDim solutions outperform all 6 MetaESBB solutions.

In Table G.6, we compare the 6 solutions of MetaESBB-OptDim to the 6 solutions of MetaESBB-RanDim when using the detailed metamodel. The table format is the same as the format of Table G.3. In Table G.3, there are 6 values in bold, which means that we cannot reject the null hypothesis of the 6 t-tests.

Table G.4: Final solution simulation: MetaESBB-OptDim, MetaESBB-RanDim and MetaESBB (detailed metamodel)

Algorithm	Run	Mean	Standard deviation
MetaESBB-OptDim	1	295850.08	375.90
	2	296213.23	333.28
	3	296056.48	406.12
	4	296251.71	328.51
	5	296732.88	420.44
	6	295910.48	392.52
MetaESBB	1	294607.35	420.63
	2	294491.98	351.18
	3	295117.07	387.08
	4	295006.46	444.68
	5	294534.41	350.68
	6	294126.67	354.63
MetaESBB-RanDim	1	294989.69	338.46
	2	295894.35	416.96
	3	295269.41	310.75
	4	295374.37	337.01
	5	295656.57	343.23
	6	296132.56	306.60

Table G.5: Final solution comparison: MetaESBB-OptDim vs. MetaESBB (detailed metamodel)

Run		MetaESBB					
		1	2	3	4	5	6
MetaESBB-OptDim	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table G.6: Final solution comparison: MetaESBB-OptDim vs. MetaESBB-RanDim (detailed metamodel)

Run		MetaESBB-RanDim					
		1	2	3	4	5	6
MetaESBB-OptDim	1	0.0000	**0.7108**	0.0000	0.0000	0.0042	**1.0000**
	2	0.0000	0.0000	0.0000	0.0000	0.0000	**0.1054**
	3	0.0000	0.0259	0.0000	0.0000	0.0000	**0.8534**
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0319
	5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	6	0.0000	**0.4213**	0.0000	0.0000	0.0004	**0.9989**

Bibliography

Sigrún Andradóttir. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 9(4):349–380, 1999.

Sigrún Andradóttir and Andrei A Prudius. Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing*, 21(2):193–208, 2009.

Bruce Ankenman, Barry L Nelson, and Jeremy Staum. Stochastic Kriging for simulation metamodeling. *Operations Research*, 58(2):371–382, 2010.

Milos Balać, Francesco Ciari, and Kay W Axhausen. Evaluating the influence of parking space on the quality of service and the demand for one-way carsharing: a Zürich area case study. In *TRB 95th Annual Meeting Compendium of Papers*. Transportation Research Board, 2016.

Milos Balac, Francesco Ciari, and Kay W Axhausen. Modeling the impact of parking price policy on free-floating carsharing: case study for Zürich, Switzerland. *Transportation Research Part C*, 77:207–225, 2017.

Russell R Barton and Martin Meckesheimer. Metamodel-based simulation optimization. In Shane G Henderson and Barry L Nelson, editors, *Handbooks in operations research and management science: simulation*, volume 13, pages 535–574. Elsevier, Amsterdam, Netherlands, 2006.

Henrik Becker, Francesco Ciari, and Kay W Axhausen. Comparing car-sharing schemes in Switzerland: User groups and usage patterns. *Transportation Research Part A*, 97:17–29, 2017.

Burak Boyacı, Konstantinos G Zografos, and Nikolas Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

Burak Boyacı, Konstantinos G Zografos, and Nikolas Geroliminis. An integrated optimization-simulation framework for vehicle and personnel relocations of electric carsharing systems with reservations. *Transportation Research Part B*, 95:214–237, 2017.

Georg Brandstätter, Claudio Gambella, Markus Leitner, Enrico Malaguti, Filippo Masini, Jakob Puchinger, Mario Ruthmair, and Daniele Vigo. Overview of optimization problems in electric car-sharing system design and management. In Herbert Dawid, Karl F Doerner, Gustav Feichtinger, Peter M Kort, and Andrea Seidl, editors, *Dynamic Perspectives on Managerial Decision Making: Essays in Honor of Richard F. Hartl*, volume 22, pages 441–471. Springer International Publishing, Cham, Switzerland, 2016.

Elvezia M Cepolina and Alessandro Farina. A new shared vehicle system for urban areas. *Transportation Research Part C*, 21(1):230–243, 2012.

Xiao Chen, Carolina Osorio, and Bruno Filipe Santos. Simulation-based travel time reliable signal control. *Transportation Science*, 53(2):523–544, 2019a.

Xiqun Chen, Zheng Zhu, Xiang He, and Lei Zhang. Surrogate-based optimization for solving a mixed integer network design problem. *Transportation Research Record*, 2497(1):124–134, 2015.

Zhiwei Chen, Yujie Guo, Amy L Stuart, Yu Zhang, and Xiaopeng Li. Exploring the equity performance of bike-sharing systems with disaggregated data: A story of southern Tampa. *Transportation Research Part A*, 130:529–545, 2019b.

Stephen E Chick and Koichiro Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49(5):732–743, 2001.

Virot Chiraphadhanakul. *Large-scale analytics and optimization in urban transportation: Improving public transit and its integration with vehicle-sharing services*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2013.

Linsen Chong and Carolina Osorio. A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. *Transportation Science*, 52(3): 637–656, 2017.

Francesco Ciari, Michael Balmer, and Kay W Axhausen. Concepts for large-scale carsharing system: Modeling and evaluation with agent-based approach. In *TRB 88th Annual Meeting Compendium of Papers*. Transportation Research Board, 2009.

Francesco Ciari, Nadine Schuessler, and Kay W Axhausen. Estimation of carsharing demand using an activity-based microsimulation approach: model discussion and some results. *International Journal of Sustainable Transportation*, 7(1):70–84, 2013.

Francesco Ciari, Benno Bock, and Michael Balmer. Modeling station-based and free-floating carsharing demand: test case study for Berlin. *Transportation Research Record*, 2416(1):37–47, 2014.

Francesco Ciari, Milos Balac, and Michael Balmer. Modelling the effect of different pricing schemes on free-floating carsharing travel demand: a test case for Zürich, Switzerland. *Transportation*, 42(3):413–433, 2015.

Francesco Ciari, Milos Balac, and Kay W Axhausen. Modeling carsharing with the agent-based simulation MATSim: state of the art, applications, and future developments. *Transportation Research Record*, 2564(1):14–20, 2016a.

Francesco Ciari, Claude Weis, and Milos Balac. Evaluating the influence of carsharing stations' location on potential membership: a Swiss case study. *EURO Journal on Transportation and Logistics*, 5(3):345–369, 2016b.

Marie-Hélène Coll, Marie-Hélène Vandersmissen, and Marius Thériault. Modeling spatio-temporal diffusion of carsharing membership in Québec City. *Journal of Transport Geography*, 38:22–37, 2014.

Gonçalo Homem Almeida Correia and António Pais Antunes. Optimization approach to depot location and trip selection in one-way carsharing systems. *Transportation Research Part E*, 48(1):233–247, 2012.

Gonçalo Homem Almeida Correia, Diana Ramos Jorge, and David Marques Antunes. The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system: an application in Lisbon, Portugal. *Journal of Intelligent Transportation Systems*, 18(3):299–308, 2014.

Alexandre De Lorimier and Ahmed M El-Geneidy. Understanding the factors affecting vehicle usage and availability in carsharing networks: A case study of communauto carsharing system from Montréal, Canada. *International Journal of Sustainable Transportation*, 7(1):35–51, 2013.

Yinghan Deng. *Design and management of mobility-on-demand (MOD) transportation systems considering demand uncertainty and flexibility - a simulation-based approach*. PhD thesis, National University of Singapore, Singapore, 2015.

Michael Duncan. The cost saving potential of carsharing in a US context. *Transportation*, 38(2):363–382, 2011.

Evan Fields, Carolina Osorio, and Tianli Zhou. A data-driven car sharing simulator for inferring latent demand. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2017. Available at: `http://web.mit.edu/osorioc/www/papers/fields17Sim.pdf`.

Evan Fields, Carolina Osorio, and Tianli Zhou. A data-driven method for reconstructing a distribution from a truncated sample with an application to inferring car-sharing demand. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2018. Available at: `https://web.mit.edu/osorioc/www/papers/fieOsoZho_carShareDem.pdf`.

Jörg Firnkorn and Martin Müller. What will be the environmental effects of new free-floating car-sharing systems? The case of Car2go in Ulm. *Ecological Economics*, 70(8):1519–1528, 2011.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Google Maps. 315 Zipcar stations in Boston area. `https://drive.google.com/open?id=1HggY_4-dOuYU71lhnudL8WqeSJA&usp=sharing`, 2017a. Accessed: 2017-09-22.

Google Maps. Zipcar stations in Boston South End neighborhood. `https://drive.google.com/open?id=1hOvbRIjfZJjF5L3OfoAThiq0_p8&usp=sharing`, 2017b. Accessed: 2017-09-22.

Google Maps. 144 Zipcar stations in Cambridge-Somerville-Medford area. `https://drive.google.com/open?id=1ZHld3f4B7DhmGwMCyDhKN0m6zWszzhx4&usp=sharing`, 2019. Accessed: 2019-11-02.

Adam Greenhall. Experimentation in a ridesharing marketplace. `https://eng.lyft.com/https-medium-com-adamgreenhall-simulating-a-ridesharing-marketplace-36007a8a31f2`, 2016. Accessed: 2018-08-15.

Long He, Ho-Yin Mak, Ying Rong, and Zuo-Jun Max Shen. Service region design for urban electric vehicle sharing systems. *Manufacturing & Service Operations Management*, 19(2):309–327, 2017.

L Jeff Hong and Barry L Nelson. Discrete optimization via simulation using COMPASS. *Operations Research*, 54(1):115–129, 2006.

L Jeff Hong, Barry L Nelson, and Jie Xu. Discrete optimization via simulation. In Michael C Fu, editor, *Handbook of simulation optimization*, volume 216, pages 9–44. Springer, New York, New York, USA, 2015.

Jiaqiao Hu, Michael C Fu, and Steven I Marcus. A model reference adaptive search method for stochastic global optimization. *Communications in Information & Systems*, 8(3):245–276, 2008.

Nanjing Jian, Daniel Freund, Holly M Wiberg, and Shane G Henderson. Simulation optimization for a large-scale bike-sharing system. In Theresa M K Roeder, Peter I Frazier, Roberto Szechtman, Enlu Zhou, Todd Huschka, and Stephen Chick, editors, *Proceedings of the 2016 Winter Simulation Conference*, pages 602–613, Piscataway, New Jersey, USA, 2016. IEEE Press.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Diana Jorge and Gonçalo Homem Almeida Correia. Carsharing systems demand estimation and defined operations: A literature review. *European Journal of Transport and Infrastructure Research*, 13(3):201–220, 2013.

Diana Jorge, Cynthia Barnhart, and Gonçalo Homem Almeida Correia. Assessing the viability of enabling a round-trip carsharing system to accept one-way trips: Application to Logan Airport in Boston. *Transportation Research Part C*, 56:359–372, 2015.

Jaeyoung Jung, Joseph YJ Chow, R Jayakrishnan, and Ji Young Park. Stochastic dynamic itinerary interception refueling location problem with queue delay for electric taxi charging stations. *Transportation Research Part C*, 40:123–142, 2014.

Jack PC Kleijnen. Kriging metamodeling in simulation: A review. *European journal of operational research*, 192(3):707–716, 2009.

Jack PC Kleijnen, Wim Van Beers, and Inneke Van Nieuwenhuyse. Constrained optimization in expensive simulation: Novel approach. *European Journal of Operational Research*, 202(1):164–174, 2010.

Mengshi Lu, Zhihao Chen, and Siqian Shen. Optimizing the profitability and quality of service in carshare systems under demand uncertainty. *Manufacturing & Service Operations Management*, 20(2):162–180, 2017.

MATSim. MATSim: multi-agent transport simulation. `https://www.matsim.org`, 2018. Accessed: 2018-08-14.

Adam Millard-Ball, Gail Murray, Jessica Ter Schure, Christine Fox, and Jon Burkhardt. Car-sharing: Where and how it succeeds. Technical Report TCRP Report 108, Transportation Research Board of the National Academies, Washington, D.C., USA, 2005.

Kalyani Shankar Nagaraj. *Stochastically constrained simulation optimization on mixed-integer spaces*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA, 2014.

Rahul Nair and Elise Miller-Hooks. Equilibrium network design of shared-vehicle systems. *European Journal of Operational Research*, 235(1):47–61, 2014.

NCSL. Car sharing - state laws and legislation. `http://www.ncsl.org/research/transportation/car-sharing-state-laws-and-legislation.aspx`, 2020. NCSL stands for National Conference of State Legislatures. Accessed: 2020-02-22.

Barry L Nelson. Optimization via simulation over discrete decision variables. In John J Hasenbein, Paul Gray, and Harvey J Greenberg, editors, *Tutorials in Operations Research: Risk and Optimization in an Uncertain World*, volume 7, pages 193–207. INFORMS, Hanover, Maryland, USA, 2010.

Vladimir I Norkin, Georg Ch Pflug, and Andrzej Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical programming*, 83(1-3): 425–450, 1998.

Eoin Daniel O'Mahony. *Smarter tools for (Citi) bike sharing*. PhD thesis, Cornell University, Ithaca, New York, USA, 2015.

Carolina Osorio. Dynamic origin-destination matrix calibration for large-scale network simulators. *Transportation Research Part C*, 98:186–206, 2019.

Carolina Osorio and Michel Bierlaire. A simulation-based optimization framework for urban transportation problems. *Operations Research*, 61(6):1333–1345, 2013.

Carolina Osorio and Linsen Chong. A computationally efficient simulation-based optimization algorithm for large-scale urban transportation problems. *Transportation Science*, 49(3):623–636, 2015.

Carolina Osorio and Kanchana Nanduri. Energy-efficient urban traffic management: A microscopic simulation-based approach. *Transportation Science*, 49(3):637–651, 2015.

Peter L Salemi. *Gaussian Markov random fields and moving least squares for metamodeling and optimization in stochastic simulation*. PhD thesis, Northwestern University, Evanston, Illinois, USA, 2014.

Peter L Salemi, Eunhye Song, Barry L Nelson, and Jeremy Staum. Gaussian Markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research*, 67(1):250–266, 2019.

Stefan Schmöller, Simone Weikl, Johannes Müller, and Klaus Bogenberger. Empirical analysis of free-floating carsharing usage: The Münich and Berlin case. *Transportation Research Part C*, 56:34–51, 2015.

Mariana Teixeira Sebastiani, Ricardo Lüders, and Keiko Verônica Ono Fonseca. Evaluating electric bus operation for a real-world BRT public transportation using simulation optimization. *IEEE Transactions on Intelligent Transportation Systems*, 17 (10):2777–2786, 2016.

Susan Shaheen and Nelson Chan. Mobility and the sharing economy: Potential to facilitate the first-and last-mile public transit connections. *Built Environment*, 42 (4):573–588, 2016.

Susan Shaheen and Adam P Cohen. Carsharing and personal vehicle services: worldwide market developments and emerging trends. *International Journal of Sustainable Transportation*, 7(1):5–34, 2013.

Leyuan Shi and Sigurdur Ólafsson. Nested partitions method for stochastic optimization. *Methodology and Computing in Applied probability*, 2(3):271–291, 2000.

Jacob Søndergaard. *Optimization using surrogate models - By the space mapping technique*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2003.

Tai Stillwater, Patricia Mokhtarian, and Susan Shaheen. Carsharing and the built environment: Geographic information system-based study of one US operator. *Transportation Research Record*, 2110(1):27–34, 2009.

Lihua Sun, L Jeff Hong, and Zhaolin Hu. Balancing exploitation and exploration in discrete optimization via simulation through a Gaussian process-based search. *Operations Research*, 62(6):1416–1438, 2014.

James R. Swisher, Sheldon H. Jacobson, and Enver Yücesan. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation*, 13(2):134–154, 2003.

Honggang Wang, Raghu Pasupathy, and Bruce W Schmeiser. Integer-ordered simulation optimization using R-SPLINE: Retrospective search with piecewise-linear interpolation and neighborhood enumeration. *ACM Transactions on Modeling and Computer Simulation*, 23(3):1–24, 2013.

Xiuxian Wang, Xuran Gong, Na Geng, Zhibin Jiang, and Liping Zhou. Metamodel-based simulation optimisation for bed allocation. *International Journal of Production Research*, pages 1–21, 2019.

Stefan M Wild, Rommel G Regis, and Christine A Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.

Jing Xie, Peter I Frazier, and Stephen E Chick. Bayesian optimization via simulation with pairwise sampling and correlated prior beliefs. *Operations Research*, 64(2): 542–559, 2016.

Jie Xu. Efficient discrete optimization via simulation using stochastic Kriging. In Christoph Laroque, Jan Himmelspach, Raghu Pasupathy, Oliver Rose, and Adelinde M Uhrmacher, editors, *Proceedings of the 2012 Winter Simulation Conference*, pages 1–12, Piscataway, New Jersey, USA, 2012. IEEE Press.

Jie Xu, Barry L. Nelson, and Jeff L. Hong. Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation*, 20(1):1–29, Feb 2010.

Jie Xu, Barry L Nelson, and L Jeff Hong. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS Journal on Computing*, 25(1):133–146, 2013.

Wendy Lu Xu and Barry L Nelson. Empirical stochastic branch-and-bound for optimization via simulation. *IIE Transactions*, 45(7):685–698, 2013.

Di Yan and H Mukai. Stochastic discrete optimization. *SIAM Journal on control and optimization*, 30(3):594–612, 1992.

Chao Zhang, Carolina Osorio, and Gunnar Flötteröd. Efficient calibration techniques for large-scale traffic simulators. *Transportation Research Part B*, 97:214–239, 2017.

Tianli Zhou. Network design for integrated vehicle-sharing and public transportation service. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2015.

Tianli Zhou, Carolina Osorio, and Evan Fields. A data-driven discrete simulation-based optimization algorithm for car-sharing service design. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2019. Available at: `http://web.mit.edu/osorioc/www/papers/zhoOsoFieCarSharing.pdf`.

Zipcar. Zipcar officially launches in Worcester, Massachusetts. `http://www.zipcar.com/press/releases/worcester`, 2017. Accessed: 2017-09-26.