A data-driven discrete simulation-based optimization algorithm for large-scale two-way car-sharing network design

Tianli Zhou
Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, tzhou90@mit.edu

Carolina Osorio
Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, osorioc@mit.edu

Evan Fields
Operations Research Center, Massachusetts Institute of Technology, efields@mit.edu

Key words: discrete simulation-based optimization, metamodel, large-scale car-sharing network design

History:

Abstract
This paper formulates a discrete simulation-based optimization (SO) algorithm for a family of large-scale car-sharing network design problems. The approach is a metamodel SO approach. A novel metamodel based on a mixed-integer program (MIP) is formulated. The metamodel is embedded within a general-purpose 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 a high-dimensional Boston case study using car-sharing reservation data. The method is benchmarked versus several algorithms. The experiments indicate that the analytical network model information, provided by the MIP, is useful both at the first iteration of the algorithm and across iterations. The solutions derived by the proposed method are benchmarked versus the solutions deployed in the field by the car-sharing operator. Via simulation, the proposed solutions outperform those deployed, both in terms of profit and vehicle utilization. This holds for all considered demand scenarios.
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 of 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. We view this general idea of combining analytical MIP formulations with general-purpose SO algorithms, or more broadly with general-purpose sampling strategies of high-resolution data, as an innovative and promising area of future research.

1. Introduction

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, Ciari, and Axhausen 2017), 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, one-way station-based, free-floating and peer-to-peer. For full definitions, see, for instance, Schmöller et al. (2015). This paper focuses on two-way services, which 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. The network design optimization problem studied in this paper is the optimal allocation of a fleet of vehicles to a set of stations.

Detailed reviews of vehicle-sharing studies are given in Jorge and Correia (2013), Brandstätter et al. (2016). Table 1 summarizes some of the recent vehicle-sharing network design literature. The column “Optimization” indicates whether the method is analytical, simulation-based optimization (SO) 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 vehicles (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 were not directly reported in the papers. The “Problem” column specifies the type of decisions the problem addresses.

The table indicates that most recent studies focus on one-way vehicle-sharing, while two-way vehicle-sharing has received less attention. Large-scale network instances consider problems with
<table>
<thead>
<tr>
<th>Paper</th>
<th>Optimization</th>
<th>Context</th>
<th>Case study size</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Analytical Simulation-based</td>
<td>One-way Two-way Free-floating Bike-sharing Car-sharing</td>
<td>Site Integer Continuous</td>
<td>Site location Fleet assignment Station capacity Other</td>
</tr>
<tr>
<td>Correia and Antunes (2012)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>75</td>
</tr>
<tr>
<td>Cepolina and Farina (2012)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>11</td>
</tr>
<tr>
<td>Chiraphadhanakul (2013, Chapter 4)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>27</td>
</tr>
<tr>
<td>Correia, Jorge, and Antunes (2014)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>116</td>
</tr>
<tr>
<td>Nair and Miller-Hooks (2014)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64</td>
</tr>
<tr>
<td>Boyaci, Zografos, and Geroliminis (2015)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100 All together $\sim 10^5$</td>
</tr>
<tr>
<td>Deng (2015, Chapter 5)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>8</td>
</tr>
<tr>
<td>Jorge, Barnhart, and Correia (2015)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>391</td>
</tr>
<tr>
<td>O’Mahony (2015, Chapter 3)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>300</td>
</tr>
<tr>
<td>Zhou (2015, Chapter 4)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
</tr>
<tr>
<td>Chen (2016, Chapter 4)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>9</td>
</tr>
<tr>
<td>Jian et al. (2016)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>466</td>
</tr>
<tr>
<td>He et al. (2017)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>61</td>
</tr>
<tr>
<td>This paper</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>315</td>
</tr>
</tbody>
</table>
300-500 site locations (e.g., stations). The majority of the network design literature focuses on analytical approaches. SO approaches are less common, yet are receiving increasing attention due to the availability of higher-resolution data. For vehicle-sharing problems, SO studies include Cepolina and Farina (2012), Deng (2015, Chapter 5) and Jian et al. (2016).

Studies of car-sharing demand include Millard-Ball et al. (2005), Stillwater, Mokhtarian, and Shaheen (2009), Coll, Vandersmissen, and Thériault (2014), Ciari, Weis, and Balac (2016). In the car-sharing context, sources of demand uncertainty include the spatial and temporal distribution of demand, as well as uncertainties in demand-supply interactions. For example, if there is no available car at the desired location and time, a customer may decide to: (i) not make any reservation, which leads to lost demand, or (ii) find an alternate available reservation, which leads to demand spillback (e.g., a vehicle at another station or at another time is reserved, also known as spillover). Hence, car-sharing demand-supply interactions are intricate to capture and to model, yet are critical to account for when planning and operating car-sharing services.

The most popular approach to address vehicle-sharing (both car- and bike-sharing) network design problems across all service types (two-way, one-way, floating) is the use of analytical mixed integer programs (MIP). Studies with deterministic demand include Correia and Antunes (2012), Chiraphadhanakul (2013, Chapter 4), Correia, Jorge, and Antunes (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 analytical probability distribution for the demand combined with optimization methods such as stochastic programming and robust optimization (O’Mahony 2015, Chapter 3, Chen 2016, Chapter 4, He et al. 2017). Recently studies have shown interests in accounting for the interactions between the car-sharing network and the public transportation network (Chiraphadhanakul 2013, Chapter 4, Nair and Miller-Hooks 2014, Zhou 2015, Chapter 3).

While stochastic simulators enable a more detailed modeling of demand and supply uncertainties, their use to address a network design optimization problem 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 (O’Mahony 2015, Chapter 5, Ciari, Balac, and Balmer 2015).

Studies, such as Cepolina and Farina (2012) and Deng (2015, Chapter 5), have included the simulator as part of the optimization framework and have resorted to general-purpose black-box 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 paper. Of particular notice is the large-scale bike-sharing instance studied in [Jian et al. (2016)], which considers a set of 466 stations.

There are a variety of discrete SO (SO problems with all decision variables being discrete) algorithms in the literature; recent reviews include [Nelson (2010) and Hong, Nelson, and Xu (2015)]. Discrete SO algorithms include Convergent Optimization via Most-Promising-Area Stochastic Search (COMPASS) (Hong and Nelson 2006), Adaptive Hyperbox Algorithm (AHA) (Xu, Nelson, and Hong 2013), R-SPLINE (Wang, Pasupathy, and Schmeiser 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, Nelson, and Hong 2010), as well as extension of AHA known as ISC-AHA (Xu, Nelson, and Hong 2013). Other common approaches to discrete SO problems include ranking-and-selection (R&S) techniques, such as Chick and Inoue (2001), Frazier, Powell, and Dayanik (2008). A R&S review can be found in Swisher, Jacobson, and Yücesan (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). Hence, their performance is typically illustrated with low-dimensional problems (e.g., decision variables of dimension around 20).

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). In transportation, fundamental optimization problems are naturally formulated as discrete problems. Additionally, realistic case studies quickly lead to high-dimensional instances. Hence, this paper designs a discrete SO algorithm suitable for high-dimensional problems and capable of identifying solutions with good performance fast (i.e., with few simulation observations available). The latter is what we refer to as computationally efficient algorithms. The work of [Xu, Nelson, and Hong (2013)] reported experiments where AHA had good performance with problems of up to 100 decision variables. Given this evidence, we use AHA as the main algorithmic building block of the method proposed in this paper.

This paper 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, Osorio, and Flötteröd (2017), Osorio, Chen, and Santos forthcoming]. 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. 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, Lüders, and Fonseca (2016), Jian et al. (2016), Boyacı, Zografos, and Geroliminis (2017).

In this paper, we consider an SO approach to address a car-sharing network design problem. We use a car-sharing network simulator (Fields, Osorio, and Zhou 2017), which relies on few demand modeling assumptions. Instead, it relies primary on sampling from high-resolution (or disaggregate) car-sharing reservation data. The simulator accounts for the intricate spatial-temporal distribution of demand as well as the intricate demand-supply interactions.

This paper proposes a metamodel SO algorithm. More specifically, we formulate a novel metamodel for this family of problems. We then combine the metamodel with an existing discrete SO algorithm, leading to a novel metamodel SO algorithm for discrete network design problems. Since the simulator exploits the high-resolution information in the reservation data, we refer to our method as a data-driven SO algorithm. In other words, the data is not merely used to fit parametric distributions, rather we iteratively sample from the data to obtain individual realizations of reservations.

The contributions of this paper can be summarized as follows. 

**Data-driven technique** As described above, the demand-supply interactions of a car-sharing system are intricate. For instance, car-sharing stations typically have low capacity (e.g., a handful of vehicles are assigned to each station). Hence, the desired reservation of a user is often not available. Thus, demand spillback (to adjacent stations or adjacent time intervals) and demand loss (users not reserving a car at all due to their desired reservation not being available) often occur. The traditional approach to car-sharing network design problems has been the analytical formulation of these demand-supply interactions. This comes at the cost of a simplified description of these intricate demand distributions and demand-supply interactions. Nevertheless, car-sharing operators collect rich and abundant data about their users and their interaction with the system. Hence, in this work, our goal is to acknowledge both the complexity of a car sharing service, 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.

To achieve this our approach is to use a sampling strategy to sample from high-resolution reservation data. The sampling strategy is based on limited modeling assumptions, yet captures the distinction between realized demand (reservations actually made, i.e., data available) and latent demand (set of reservations that could have been made, regardless of whether or not they were made). For a detailed description of the distinction between these two types of demand, see Fields, Osorio, and Zhou (2017). It is this sampling strategy, which we call the *simulator*. In other words, a simulation run consists of sampling from the car-sharing data such as to simulate how latent demand leads to actual car reservations. This paper designs an algorithm that enables us to sample
from this rich dataset to address intricate (high-dimensional, stochastic, mixed-integer) network design and operations problems. It is therefore a data-driven technique. Most past work in network design for car-sharing studies has limited the use of field data to fitting the parameters of an analytical MIP model. This fitting is typically done once prior to solving the MIP. In this paper, we go beyond this: the data is used at every iteration of the algorithm to provide a more accurate estimation of the performance of a given network design. The information captured in the data about the intricate demand distribution and demand-supply interactions is exploited at every iteration of the algorithm. The use of a data sampling strategy allows us to overcome the need (and major challenge) of analytically modeling the demand distribution or the demand-supply interactions. To the best of our knowledge, this is the first work to design an algorithm that both relies on high-resolution data and preserves this high-resolution (i.e., does not merely aggregate the disaggregate data) for car-sharing network design optimization.

**High dimensional discrete SO problems** The proposed algorithm is suitable to address high-dimensional network design problems. In Section 3.3 we use it to address a problem 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 the scalability by formulating a scalable MIP metamodel and combining it with AHA, yielding the newly proposed discrete SO algorithm, which we call MetaAHA. The approach combines the merits of both analytical and simulation-based optimization methods.

**Computationally efficient algorithm** The proposed algorithm is designed to identify good quality solutions within few iterations (i.e., when few simulation observations available). This differs from most discrete SO literature which is typically 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. The results of Section 3 indicate that this analytical structural information is the key to achieving computational efficiency. Moreover, as is illustrated with the experiments of Section 3 the proposed metamodel can be combined with a variety of general-purpose discrete SO algorithms to identify good initial solutions to the problem. This contributes to improve the short-term performance of asymptotically-designed general-purpose discrete SO algorithms.

**Metamodeling for discrete SO** The main feature of the proposed algorithm is the formulation of a novel metamodel, (i.e., an analytical approximation of the simulation-based objective function) which combines a general-purpose component (the functional metamodel) and a problem-specific
component (the physical metamodel). In particular, the physical component is a novel MIP formulation and is the key to achieving computational efficiency and robustness to the quality of the initial points. Such metamodel ideas for transportation problems have been successfully formulated for various continuous SO problems. This is the first paper that extends these ideas to the discrete SO setting. The paper shows that by using such metamodel ideas, high-dimensional discrete SO problems can be addressed in a computationally efficient way. The paper shows how the proposed metamodel ideas enable general-purpose discrete SO algorithms to become more scalable (i.e., suitable for higher-dimensional problems). Since the most fundamental OR transportation optimization problems (e.g., routing) are naturally formulated as discrete optimization problems, the ideas of this paper lay the foundations for a variety of important and difficult transportation problems to be addressed efficiently with data-driven, or simulation-based, network models.

**Boston metropolitan area case study** This work has been carried out in collaboration with private stakeholders, Ford and Zipcar, that operate car-sharing systems. We use Zipcar data in Section 3 to address two Boston case studies, one of which is a high-dimensional problem with 315 stations. This is considered high-dimensional for car-sharing network design problems. Other recent high-dimensional case studies consider between 391 and 466 stations (Jorge, Barnhart, and Correia 2015, Jian et al. 2016).

Section 2 formulates the proposed methodology. Its performance is evaluated in Section 3 with experiments on both synthetic toy network and Boston networks. Conclusions are presented in Section 4. Algorithmic details are presented in Appendix A.

## 2. Methodology

This section presents the proposed methodology. The network design problem is formulated (Section 2.1). The general metamodel SO framework is summarized (Section 2.2). The metamodel for car-sharing network design problems is formulated (Section 2.3) and the proposed algorithm is described (Section 2.4). The car-sharing network simulator used in this paper, as well as the role of the car-sharing data, is summarized in Section 2.5.

### 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);
- \( R(x, q_1) \): random variable representing the revenue;
- \( c_i \): cost, over the planning period, of a parking space at station \( i \);
- \( q_1 \): 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, \( \mathcal{I} = \{1, 2, \ldots, I\} \);
- \( F \): feasible region.

The problem is formulated as follows:

\[
\max_x g(x; q_1) = E[R(x; q_1)] - \sum_{i \in \mathcal{I}} c_i x_i
\]  

subject to

\[
\sum_{i \in \mathcal{I}} x_i \leq X  
\]
\[
x_i \leq N_i \quad \forall i \in \mathcal{I}  
\]
\[
x_i \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}.  
\]

The objective function represents the expected profit for a given fleet assignment vector, \( x \). It is defined as the difference between the expected revenue \( E[R(x; q_1)] \) and the expected costs. The expected revenue is a simulation-based function, estimates of which can be obtained via simulation. The simulator, which is described in more detail in Section 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 be satisfied. In other words, realizations of the revenue random variable \( R \) are obtained by sampling from car-sharing reservation data. The expected cost is linear in the number of parking spots used at station \( i \). The cost parameters, \( c_i \), are exogenous. In this work, they represent parking space leasing fees. Constraint (2) bounds the total number of cars assigned across all stations with the fleet size. Constraint (3) 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 (4)). Constraints (2)-(4) specify the feasible region, \( F \).

Problem (1)-(4) consists of a simulation-based objective function with discrete variables and analytical (i.e., non-simulation-based) constraints. The main challenges of addressing this problem are the following. There is no analytical expression available for the objective function, hence traditional (analytical) discrete optimization algorithms cannot be used. Discrete SO problems
inherit the curse of dimensionality of discrete analytical problems. The function \( g \) is often an intricate function with several local optima. Given the simulation-based nature of the objective function, we cannot observe \( g \), rather we can only estimate it by running a set of simulation replications. Hence, practitioners often terminate the SO algorithm prior to convergence, rather than waiting until convergence tests are passed.

Given the challenges of addressing Problem 1)-(4), we propose an algorithm that at every iteration of the algorithm, 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.

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

\[
\max_x m_k(x; \beta_k, q_2) = \beta_{k,0}g_A(x, z; q_2) + \phi(x; \beta_k) \quad (5)
\]

\[
h(x, z; q_2) = 0 \quad (6)
\]

\[
x \in \mathcal{F}, \quad (7)
\]

where \( m_k \) is known as the metamodel, \( \beta_k \) is a vector of metamodel parameters with \( i^{th} \) element \( \beta_{k,i} \), \( z \) is a vector of additional endogenous variables, \( q_2 \) is a vector of exogenous parameters, \( g_A \) is the approximation of \( g \) (Equation 1) derived by the analytical network model and \( \phi \) is a polynomial. The analytical network model is denoted as \( h \) (Equation 6). It is defined here as a system of equations, which yields a problem-specific approximate and analytical mapping between the decision vector \( x \) and the simulation-based objective function \( g \) (Equation 1), which in this paper represents the expected profit.

The metamodel optimization problem (i.e., the above problem (5)-(7)) differs from Problem 1)-(4) in that: (i) it replaces the (unknown) simulation-based objective function \( g \) of Equation 1) with an analytical function \( m_k \); (ii) it has additional constraints (Equation 6), which will be formulated and discussed in Section 2.3. 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 \( g(x) \). 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 \( g(x) \), which as
will be discussed in Section 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 already simulated points).

Metamodels are typically classified as either functional metamodels or physical metamodels (Søndergaard 2003, Chaper 2). The former are general-purpose functions chosen based on their mathematical properties (e.g., low-order polynomial functions, radial-basis functions, Kriging functions). Functional metamodels have been commonly used for both continuous SO (Jones, Schonlau, and Welch 1998, Barton and Meckesheimer 2006, Wild, Regis, and Shoemaker 2008, Kleijnen, Van Beers, and Van Nieuwenhuyse 2010, Ankenman, Nelson, and Staum 2010) and for discrete SO (Xu 2012, Sun, Hong, and Hu 2014, Salemi 2014 Chapter 4, Xie, Frazier, and Chick 2016). Physical metamodels are problem-specific functions that attempt to capture some structure of the underlying problem. The main idea of the formulation of Equation (5) is to combine a physical problem-specific component \( g_A \) term with a functional general-purpose component \( \phi \) term. The metamodel is a parametric function, with parameter vector \( \beta_k \), which are fitted at every iteration of the SO algorithm based on simulated observations. More specifically, the parameters are fit such as to minimize a distance function between \( m_k(x) \) and the set of objective function estimates available at iteration \( k \), \( \hat{g}(x) \). For more details on the fitting of the metamodel, see Appendix A. To the best of our knowledge, this is the first work to consider a metamodel with both a problem-specific physical component and a general-purpose component for discrete SO problems.

In this paper, we formulate \( g_A \) as the objective function of an analytical mixed-integer linear fleet assignment problem, and \( \phi \) is defined as a linear function of \( x \):

\[
\phi(x; \beta_k) = \beta_{k,1} + \sum_{i \in I} \beta_{k,i+1} x_i. \tag{8}
\]

Hence, Problem (5)-(7) is formulated as a mixed-integer linear program.

In summary, the metamodel consists of an analytical problem-specific objective function \( g_A \) which is corrected parametrically by a scaling term \( \beta_k \) and an additive linear error term \( \phi \). 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.
2.3. Car-sharing network design metamodel formulation

To formulate the analytical physical component of the metamodel, \( g_A \), which approximates the profit of a given network design strategy, we introduce the following additional notation.

- \( d_{it} \): number of customers that desire a reservation at station \( i \) with start time \( t \) and duration \( l \);
- \( r_{it} \): 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 for) station \( j \);
- \( z_{it} \): number of customers that make a reservation at station \( i \) with start time \( t \) and duration \( l \);
- \( z_{ji} \): 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 \( l \);
- \( z \): vector that combines all variables \( \{z_{it}\} \) and \( \{z_{ji}\} \);
- \( t_{\text{max}} \): number of one-hour reservation start time intervals during the planning period (e.g., for an \( n \)-day planning period, \( t_{\text{max}} = n \times 24 \));
- \( \mathcal{I}_i \): set of stations “near” station \( i \), including station \( i \);
- \( \mathcal{L} \): set of reservation durations (in hours), \( \mathcal{L} = \{1,2,\ldots,24\} \);
- \( \mathcal{T} \): set of reservation start time interval indices, \( \mathcal{T} = \{1,2,\ldots,t_{\text{max}}\} \);
- \( \mathcal{T}_i(t,l) \): 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 function \( g_A \) is formulated as:

\[
g_A(x, z) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} \sum_{t \in \mathcal{T}} \sum_{l \in \mathcal{L}} p^{ij} r_{it} z_{ji} - \sum_{i \in \mathcal{I}} c_i x_i, \tag{9}
\]

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. Note that demand spillback and loss are described in a more detailed and disaggregate manner in the simulator (see Section 2.5).

The auxiliary variable \( z_{ji} \) is related to the decision vector \( x \) through the analytical network model, which is denoted with \( h \) in Equation (6) and is defined as follows.

\[
\sum_{j \in \mathcal{I}_i} z_{ji} = z_{it} \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{10}
\]

\[
\sum_{j \in \mathcal{I}_i} z_{ji} \leq d_{it} \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{11}
\]

\[
\sum_{l \in \mathcal{L}} z_{it} + \sum_{l \in \mathcal{L}} \sum_{t' \in \mathcal{T}_i(t,l)} z_{lt} \leq x_i \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{12}
\]

\[
z_{it} \in \mathbb{R}_+ \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \forall l \in \mathcal{L} \tag{13}
\]

\[
z_{ji} \in \mathbb{R}_+ \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{I}_i, \forall t \in \mathcal{T}, \forall l \in \mathcal{L}, \tag{14}
\]
where $\mathcal{T}_i(t, l) = \{ t' \in \mathcal{T} : t' + 1 \leq t \leq t' + l - 1 \}$. Equation (10) states that $z_{it}$, 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 $i \in \mathcal{I}$, hence this summation includes the reservations that were desired and also made at station $i$ (with start time $t$ and duration $l$). Equation (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 (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 (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 $i$. Equations (13) and (14) assumes non-negative real values for the auxiliary variables ($z_{it}$ and $z_{ij}$). 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 $d_{it}$, $r_{it}$, $c_i$ and $p^{ij}$, while the endogenous variables are $z_{it}$, $z_{ij}$ and $x_i$. The exogenous parameters $r_{it}$, $c_i$ and $p^{ij}$ are directly estimated from the data and in consultation with Zipcar staff. The demand parameters ($d_{it}$) are estimated by running one demand sampling step of the simulator, as defined in Section 2.5. A discussion on the simplifications of this analytical model compared to the simulator are given in Section 2.5.

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 $O(W |\mathcal{I}| |\mathcal{T}| |\mathcal{L}|)$, and the number of constraints is in the order of $O(|\mathcal{I}| |\mathcal{T}| |\mathcal{L}|)$. 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 3.2, the MIP (5)-(7) is solved on average in 3.1 seconds for the the Boston South End network and in 62.8 seconds for the large-scale Boston network.
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, Nelson, and Hong (2013). As discussed in Section 1, AHA has been used to solve high-dimensional problems with a decision vector of dimension 100. We use the term current iterate to denote, at a given iteration, the point considered to have best performance. The name AHA stems from the sampling, at every iteration, from a region which is the intersection of the feasible region and the hyperbox (See Appendix A). 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. Denote \( \mathcal{H}_k \) as the hyperbox of iteration \( k \). The proposed algorithm, denoted MetaAHA, is an extension of AHA. Algorithm 1 presents the proposed MetaAHA algorithm.

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 are given in Appendix A.

Algorithm AHA is be 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. They identify points derived using problem-specific analytical structural information provided by the analytical network design model. This 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) to become less sensitive to the quality of the initial sample. This sensitivity to the quality of the initial sample has been identified and discussed in past AHA work (Xu, Nelson, and Hong 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 this paper in Appendix A), 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 the algorithm 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
Algorithm 1 MetaAHA

Initialization: Set $k = 1$. Let $\mathcal{H}_1 = \{x : 0 \leq x_i \leq \tilde{N}_i, \forall i \in I\}$.

1. **Identify the set of $n$ points to sample**
   
   (a) Obtain $r$ points in $\mathcal{F} \cap \mathcal{H}_k$ based on the asymptotically uniform sampling mechanism of AHA.
   
   (b) Obtain 1 point, denoted $x_{k}^{\text{meta}}$, as the solution to the metamodel optimization problem (5)-(7).
   
   (c) Obtain 1 point, denoted $x_{k}^{\text{meta-hyper}}$, as the solution to the metamodel optimization problem (5)-(7) with the additional constraint that the point belong to $\mathcal{H}_k$.

2. **Simulation**

   Simulate current iterate and the points identified in Step 1. Select the point with best performance (i.e., update the current iterate). Update the hyperbox.

3. **Check for algorithm termination**

   (a) Test if the current iterate is a local optimum as in AHA. If so, stop.
   
   (b) If the total number of iterations exceeds the maximum number of iterations (i.e., the computational budget is depleted), stop.

4. **Metamodel update**

   (a) For any simulated point, $x$, that has not been evaluated by the analytical network model, evaluate it (i.e., for a given $x$, maximize $g_A(x, z)$ of Equation (9) over $z$ subject to Constraints (10)-(14)).
   
   (b) Use all simulation observations collected so far to fit the metamodel parameters, $\beta_k$, i.e., solve the least squares problem given in Appendix A.

5. **Update iteration counter**

   Start a new iteration ($k = k + 1$), proceed to Step 1.

Based on the use of the metamodel. This 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 potential to improve the performance of these general-purpose algorithms.

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, Osorio, and Zhou (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. 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) that arise from the (unknown) latent demand distribution. This allows distinguishing 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 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). 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 is known as first-come-first-reserve. 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 costs (i.e., price paid per reservation). 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 few parameters, which are estimated from the data. Additionally, there are very few modeling assumptions. The few assumptions 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 (i.e., demand sampling and reservation simulation) of the simulation process described above 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 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 a highly 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. The notions of demand spillback are approximated through the discounted revenue parameter, $p_{ij}$ of Eq. (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 and bounds above the reservation duration to 24 hours. Based on the Boston dataset, reservations longer than 24 hours are not common.

3. Case studies

In this section, we apply the MetaAHA algorithm to optimize the design of two-way car-sharing systems. Section 3.1 considers a low-dimensional problem with synthetic toy networks. Sections 3.2-3.3 consider high-dimensional problems for Zipcar networks in the city of Boston. Zipcar is the world’s largest car-sharing service 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. The majority of its operations are two-way service. We study its two-way services for two Boston areas: (i) an area of downtown Boston known as South End (Section 3.2) and 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 3.3).

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 (9)) provided by the analytical network model of the simulation-based objective function ($g$ of Equation (1)). We consider 3 networks with topologies that are simple and are representative of subnetwork topologies of Zipcar’s Boston network. 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). The 3 networks are displayed in Figure 1. Each circle represents a car-sharing station. Each network has four stations. Stations that are considered neighbors are connected with an edge. Each station has a capacity of 6 vehicles (i.e., $N^i$
of Equation (3) equals 6), the fleet size is unlimited (i.e., the space of Equation (2) 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.

![Toy network topologies](image)

**Figure 1** Toy network topologies

For each network, we generate a group of 10 demand scenarios. The use of various demand scenarios serves to account for demand stochasticity. The demand scenarios are generated through the demand sampling step which is described in Section 2.5. For a given point, \( x \), 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, \( x \), the final estimate of its simulation-based performance, \( \hat{g}(x) \), is obtained as the average over 20 simulation replications. For the analytical model, we generate a different demand scenario to estimate its exogenous parameters (\( d_{it} \) of Equation (11).) For a given point \( x \in \mathcal{F} \), the analytical objective function, \( g_A(x, z^*) \), is obtained by maximizing Equation (9) over \( z \) subject to Constraints (10)-(14).

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

![Comparison of Analytical Objective Value and Mean Simulated Objective value of Toy Networks](image)

**Figure 2** Comparison of Analytical Objective Value and Mean Simulated Objective value of Toy Networks
For all three plots, there is a positive linear correlation between the analytical approximations, $g_A(x, z^*)$, and the simulation-based estimates, $\hat{g}(x)$. This indicates that for all three representative network topologies the analytical network model provides a good approximation of the simulation-based objective function.

### 3.2. Boston South End Zipcar network

We now consider the South End neighborhood in downtown Boston. A map of the area is displayed in Figure 3. 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., $X = 101$). 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 25. At every iteration, the number of points to be simulated is set to 10 ($n = 10$).

![Figure 3 Zipcar stations in Boston South End neighborhood (map data: Google Maps (2017b))](image)

To account for the stochasticity of demand, we proceed as in Section 3.1. We consider a group of 10 demand scenarios. For a given point, $x$, 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 4 contains four plots. Each plot considers a different group of 10 demand scenarios.
As in Section 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. 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. 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 (9) over both \( x \) and \( z \) subject to Constraints (2)-(4) and (10)-(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. To limit the premature convergence of AHA, [Xu, Nelson, and Hong (2013)] have combined it with the multi-start ISC framework ([Xu, Nelson, and Hong 2010]). Second, as the iterations advance, AHA identifies points with improved performance. This is consistent with the experiments and observations in [Xu, Nelson, and Hong (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 performance of the final solution derived by MetaAHA (i.e., the current iterate at the final iteration) are 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 4 some lines terminate prior to iteration 25. This occurs if a current iterate is considered to be a local optimum (as is indicated in Step 3a of Algorithm 1). 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, Osorio, and Flötteröd 2017], [Chong and Osorio 2017], [Osorio, Chen, and Santos forthcoming]).

The results of Figure 4 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 (9) over both $x$ and $z$ subject to Constraints (2)-(4) and (10)-(14)), and then to use the traditional AHA 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 4. Figure 5 display four plots. Each plot considers a given set 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 following trends are common to the four plots. First, MetaAHA outperforms AHAInit across most iterations. This indicates the added value of combining the analytical network design information $g_A$ with the simulation information leading to the metamodel $m$. 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 to using the analytical information across iterations. Second, AHAInit tends to converge quickly to a local (non-global) optimum. It does not deplete its computational budget. Often, this local optimum has performance that is similar to that of the point obtained by solving the analytical network design problem (i.e., maximize $g_A$ subject to Constraints (2)-(4) and (10)-(14)).

For the 12 runs MetaAHA of Figure 5 (i.e., 3 runs for each of the 4 plots), there are a total of 97 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 $x^{\text{meta}}$), (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 \( x^{\text{meta-hyper}} \)), or (iii) it can be obtained from random sampling (i.e., Step 1a of Algorithm 1, which yields points denoted \( x^{\text{sampled}} \)). Note that a point can be both of type \( x^{\text{meta}} \) and of type \( x^{\text{meta-hyper}} \). This occurs when the solution to the metamodel optimization problem in the entire feasible region is located in the hyperbox. Of the 97 different current iterates of the 12 MetaAHA runs in Figure 5, more than two thirds (i.e., 69.1% or 67 points) are of type \( x^{\text{meta}} \) or \( x^{\text{meta-hyper}} \), while less than one third (30.9% or 30 points are of type \( x^{\text{sampled}} \)). 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 2136 points. Only 18% of the simulated points are obtained by solving the metamodel (389 points), while the remaining 82% are obtained by random sampling. Hence, even though the points derived by metamodel evaluations represent only 18% of the total set of sampled points, they lead to 75% of the final solutions and 69% of the current iterates. This highlights the added value of the structural information provided by the analytical MIP. Among the 67 current iterates obtained by using structural analytical information, 22 are of type \( x^{\text{meta}} \) and 54 are of type \( x^{\text{meta-hyper}} \) (note that 9 points are both of type \( x^{\text{meta}} \) and \( x^{\text{meta-hyper}} \)). This shows that both the global (i.e., in the entire feasible region) and the local information of the analytical network model serves 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 6 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 5), we estimate the average (over the 50 demand scenarios, each scenario is simulated with 20 replications) performance. The proposed solution is that with the best (i.e., largest) average performance.

We now compare the performance of the two fleet assignments: proposed assignment and Zipcar assignment. Figure 6 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...
Figure 6  Comparison of the Zipcar fleet assignment with the proposed assignment

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 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. In other words, both profit and vehicle utilization are improved. 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.

3.3. Boston city network

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 7 displays the stations as red circles. We consider the same planning period as before. The station capacity, $N^i$, is set to 16. 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 set 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 ($K = 40$) and the number of points to simulate per iteration to 70 ($n = 70$).

Figure 8 displays the results of 4 MetaAHA runs (solid lines) and 4 AHAInit runs (dashed lines). Only 1 of the 8 runs depletes the computational budget (i.e., stops at iteration 40). It corresponds to a MetaAHA run. More specifically, the four MetaAha runs stop at iterations 11, 18, 31 and 40.
Those of AHAInit stop at iterations 12, 15, 16 and 19. All four runs of AHAInit yield final solutions with similar objective estimates. These are similar to two of the MetaAHA runs. The remaining two MetaAHA runs yield the solutions with the best performance estimates.

Figure 8 compares the performance of the best solution identified by MetaAHA with the fleet assignment strategy used by Zipcar. The best strategy proposed by MetaAHA is that with the largest estimated objective function across the 4 runs of Figure 8. To evaluate the performance of a
given fleet assignment strategy (proposed or Zipcar), we proceed as before. We generate 50 demand scenarios. For each of the 4 final solutions derived by MetaAHA and for each demand scenario, we run 20 simulation replications to estimate the average profit per solution. The solution with the highest average simulated profit is selected as the proposed solution. Figure 9 compares the performance of the proposed solution to that of Zipcar. It 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 50 demand scenarios. Each point estimate is displayed along with a 95% confidence along both directions. The confidence intervals are computed based on 50 replications. For both plots, the 50 points, which represent 50 different demand scenarios, are above the diagonal. This indicates that 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.

![Figure 9](image-url) Comparison of the Zipcar fleet assignment with the proposed assignment

### 4. Conclusions

This paper 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 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. One of the case studies is a high-dimensional
problem. The method is benchmarked versus two types of algorithms that differ only in their use of the analytical MIP information: one benchmark algorithm does not use any analytical network information (i.e., no MIP information), the second benchmark algorithm 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.

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 of 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.

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 paper 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 the 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 paper illustrates how the scalability, computational efficiency and robustness of these SO algorithms can be enhanced such as to enable it to address realistic transportation problems.

Acknowledgments

This research is part of the project “Car-sharing services: optimal network expansions to integrate automotive with mass-transit systems and to mitigate congestion at the metropolitan scale” of the Ford-MIT Alliance. The authors thank the Ford project members, including Perry MacNeille and Paul Beer. The authors also thank Arvind Kannan (Zipcar) for his valuable feedback and the Zipcar staff for the data.
Appendix A: 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 \( x \) up until and including iteration \( k \), denoted \( N_k(x) \), is computed based on the approach of AHA (Xu, Nelson, and Hong 2013). It is given by \( N_k(x) = \min\{5, \lceil 5(\log k)^{1.01} \rceil \} \). If at a given iteration \( k \), the number of simulation replications of point \( x \) obtained from previous iterations is greater or equal to \( N_k(x) \), then we do not evaluate additional replications.

In Step 2 of the algorithm, the hyperbox is updated based on the AHA approach (Xu, Nelson, and Hong 2013). More specifically, the hyperbox is defined (or updated) at iteration \( k \) as \( H_k = \{ x : l^k_i \leq x_i \leq u^k_i, \forall i \in I \} \). The bounds \( l^k_i \) and \( u^k_i \) are defined as follows. Let \( x^k \) denote the current iterate and \( E(k) \) denote the set of points that have been simulated up until and including iteration \( k \). For each \( i \in I \),

\[
l^k_i = \max_{x \in E(k), x \neq x^k} \{ x_i : x_i < x^k_i \}
\]

if \( l^k_i \) is empty, then set \( l^k_i = 0 \). Similarly,

\[
u^k_i = \min_{x \in E(k), x \neq x^k} \{ x_i : x_i > x^k_i \}
\]

if \( u^k_i \) is empty, then set \( u^k_i = N^i \).

Step 4b of the algorithm fits the metamodel parameters, \( \beta_k \), by solving the below least squares problem, which is formulated and discussed in more detail in Osorio and Bierlaire (2013).

\[
\min_{\beta} \sum_{x \in E(k)} [w_k(x) (\hat{g}(x; q_1) - m_k(x; \beta_k, q_2))^2 + (w_0(\beta_k, 0 - 1))^2 + \sum_{i=1}^{\lvert I \rvert + 1} (w_0 \beta_{k,i})^2],
\]

where \( w_0 \) is a fixed scalar weight, \( \hat{g}(x; q_1) \) represents the simulated estimate of the profit function for point \( x \), and the weight \( w_k(x) \) function is defined as \( w_k(x) = 1/(1 + \| x - x^k \|_2) \). The least squares problem minimizes a weighted distance between the simulated profit estimates \( \hat{g} \) and the metamodel predictions \( m_k \). Each point is weighted by a distance function that gives more weights to points that are closer to the current iterate, such as to improve the local (i.e., close to the current iterate) fit of the metamodel. The additional terms in the least squares problem are included such as to ensure a full rank least-squares matrix.

References


Google Maps, 2017a 315 Zipcar stations in Boston area. https://drive.google.com/open?id=1HggY_4-dOuYU71hnu4L8wqeq&usp=sharing


