Locating Parking Hubs in Free-Floating Ride Share Systems via Data-Driven Optimization

Anmar Arif, *Member, IEEE,* Kostas Margellos, *Member, IEEE*

***Abstract*—This paper presents a data-driven study on locating parking hubs in free-floating ride share systems. Recently, there has been an increase in free-floating ride share systems, where users are allowed to pick up and drop off shared vehicles anywhere in the service area. However, these systems can suffer from significant demand and supply imbalance, while certain parking habits may disturb the desired city layout. A potential solution is to allocate parking hubs in an optimal manner to regulate the behaviour of the users. This paper develops a scenario optimization model for finding the optimal locations of parking hubs. The model determines the capacities and locations of the parking hubs, while considering the uncertainty of parking demand and points of interest in the area. We design an algorithm that combines the idea of Constraint-and-Column Generation and the Alternating Direction Method of Multipliers (ADMM) algorithm to solve the optimization problem in a decentralized manner, and accompany the computed solution with a probabilistic performance certificate. We also compare the adopted approach with respect to a worst case paradigm both in terms of computational cost and in terms of conservatism of the resulting solution. Numerical results show that the proposed method leads to a less conservative performance compared to the worst case method, and reduces the computational cost compared to the classical ADMM.**

***Index Terms*—Data-driven optimization, facility location, ride share, integer programming, robust optimization.**

NOMENCLATURE

Sets and Indices

i/j	Index for grid cell
l	Index representing the type of point of interest
t	Index for time step
$\bar{G}$	Set of all grid cells
G	Set of grid cells with average departure more than 1
$G(i)$	Set including cell i and cells adjacent to i
I	Set of point of interest types
I^G	Set of grid cells with points of interests (POIs) and large daily departure
T_d	Set of time steps per day

Parameters

A_{it}	Number of bikes arriving at grid cell i and time t
B	Total number of bikes
$\bar{C}_i/\underline{C}_i$	Maximum/minimum capacity of cell i
C^h	Cost of establishing a parking hub
C^b	\$/bike-space cost
P_{it}	Number of departed bikes from grid cell i and time t
D_i	Average daily departures for cell i
F_i^0	Initial number of bikes at grid cell i
L_{il}	Number of POI of type l at grid cell i
Y	Maximum number of bikes allowed to be transferred from their preferred location
δ_i	Parking demand of grid cell i

Decision Variables

f_{it}	Parking demand of grid cell i at time t
r_{it}	Number of bikes moved from/to grid cell i
u_i	Capacity of the parking hub at i
x_i	Binary variable equals 1 if cell i has a parking hub.
y_{ij}	Parking demand shifted from i to j.

A. Arif is with the Department of Electrical Engineering, King Saud University, Riyadh, Saudi Arabia. (E-mail: anarif@ksu.edu.sa)

K. Margellos is with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, United Kingdom, (E-mail: kostas.margellos@eng.ox.ac.uk)

I. INTRODUCTION

SEVERAL cities around the world have deployed electric bike or scooter sharing systems in the past years [1]. Scooter and bike sharing systems (BSSs) offer electric scooters and bikes for short-term renting, providing consumers with a fast and convenient mode of transportation for short distances. Here we focus on bike sharing systems, but our algorithms are applicable to other shared systems as well. These types of systems are becoming increasingly popular as they are easy to park and reduce traffic congestion and air pollution. Many of the BSSs use docking stations where vehicles are locked to computer-controlled racks and users unlock the vehicle by entering their payment information. Users then must park the bike or scooter to docking stations belonging to the same system at the end of their journeys.

In recent years, a new system called dockless or free-floating bike-sharing system (FFBS) has gained popularity. In FFBS, users locate and unlock electric bikes using their smartphones and drop off bikes at their desired destinations. The idea of FFBS is to remove fixed stations and avoid their associated costs. The main advantages of FFBS are: (1) increased convenience for users as they can pick up and drop off bikes from and at any location; (2) reduced system start-up and maintenance costs since docking stations and kiosk machines are removed; (3) issues related to full parking stations are no longer present. However, similarly to station-based ride sharing systems, the distribution of bikes in FFBS can be unbalanced and the availability of bikes at desired pick up locations cannot be guaranteed. Since consumers are allowed to leave bikes anywhere, the bikes can become scattered all over the service territory. FFBS have also caused inconvenience to pedestrians due to parking on sidewalks, roads, and entryways. Thus, there is a need to regulate the drop-off and parking process of FFBS. A possible approach is to introduce parking areas/hubs/virtual stations where users can drop off the rented bicycles, as Nice Ride Minnesota implemented for their ride share system [2]. The

company allows users to park outside of the parking hubs, but with a certain convenience fee; a similar approach is also implemented by Beryl Bikes [3].

Earlier research on ride share systems focuses on BSS. In particular, research has been conducted on rebalancing [4], demand prediction [5], and optimal locations of docking stations [6]. The study in [7] presented various mixed integer programming models for solving the bike redistribution problem in station-based BSSs. The authors in [8] developed a mixed-integer linear programming (MILP) model for designing a bike sharing system, which includes determining the locations of BSS stations, number of docking stations, and fleet size.

For FFBS, the authors in [9] presented a study on using incentives for rebalancing bikes in dockless systems. The paper in [10] presented a study for optimizing the distribution and location of geofence sites (virtual geographic areas used to restrict bike drop off). However, less activity has been noticed on parking allocation for FFBS. The problem of optimizing the location of parking hubs can be classified as a facility location problem. This class of problems has been investigated extensively in a wide variety of fields and applications, such as public parking allocation [11] and allocating emergency response units [12]. For locating parking hubs in FFBS, a mixed integer linear program is designed to optimize the locations of virtual parking hubs in [13]. The MILP model was solved using a clustering algorithm and implemented on a subregion in Beijing. The paper assumed that each starting point of a journey is a potential virtual station. The proposed method assigned virtual stations at different locations during the day, so that users park the bikes at locations with higher demand. In [14], the authors used the K-means clustering algorithm to decide the location of virtual stations. The paper clustered the starting locations of users' trips, and set each cluster to be a parking hub similar to [13]. However, K-means identified some remote bikes as parking hubs, which is expected since K-means assumes all variables have the same importance for each cluster and the only consideration refers to distance. In [15], the authors used a heuristic method based on the analytic hierarchy process and the weight-restricted data envelopment analysis to select the location of parking spots.

The facility location problem is NP-hard and is thus difficult to solve in polynomial time [16]. Adding uncertainty to the problem makes it even more difficult, and standard commercial solvers cannot directly solve the problem. In this paper, we develop a data-driven scenario optimization method for locating parking hubs in FFBS with parking demand being uncertain. The objective of the optimization problem is to minimize the costs of parking hubs and parking spaces. We model uncertainty by means of scenarios and formulate a so called scenario program. To solve the resulting problem in an efficient manner, we design an algorithm that combines the Alternating Direction Method of Multipliers (ADMM) [17] and Constraint-and-Column (CC) Generation [18] for solving the parking hub allocation problem. Moreover, we accompany the resulting solution with probabilistic feasibility certificates. We refer to this algorithm hereinafter as CC-ADMM. The presented method is tested on Mobike's [19] large-scale free-floating system in Beijing. The contributions of this paper are summarized as follows:

1) We develop a mathematical model for optimizing the locations of parking hubs with uncertain parking demand in free-floating ride share systems.
2) We design an algorithm combining ADMM with Constraint-and-Column Generation for solving large-scale scenario optimization instances of such problems.
3) We show how to accompany the solution with a certificate on the probability that it remains feasible when a new uncertainty realization is encountered.

The rest of the paper is organized as follows: Section II presents the data used in this study. Section III develops models for optimizing the locations of parking hubs and Section IV presents the CC-ADMM algorithm. In Section V we discuss the (probabilistic) robustness properties of the resulting solution. Numerical results are presented in Section VI, while Section VII concludes this paper.

II. Data Description and Analysis

The data used in this study is for a FFBS system in Beijing, operated by Mobike [19]. Mobike is a bike sharing service that operates in many cities around the world. Mobike's bikes are equipped with Global Positioning System (GPS) trackers, theft prevention mechanisms, and a quick response (QR) code lock. Users unlock the bikes using a mobile application and the fee is then charged automatically depending on the trip taken by the user. Therefore, the bikes do not require docking stations and users can drop off the bikes at any location, which is a convenient service compared to traditional BSSs. The data used in this papers includes over 3 million journeys that occurred in 2017. The reported information on each journey includes: order ID, user ID, bike ID, bike type, start time, and start and end locations. There are 467,340 bikes available for the users. Fig. 1 shows 10,000 randomly selected starting points from all the journeys in the data, across a period of two weeks. The average distance taken by the riders is over 800 m. Fig. 2 shows a close up image of the region with samples of trips.

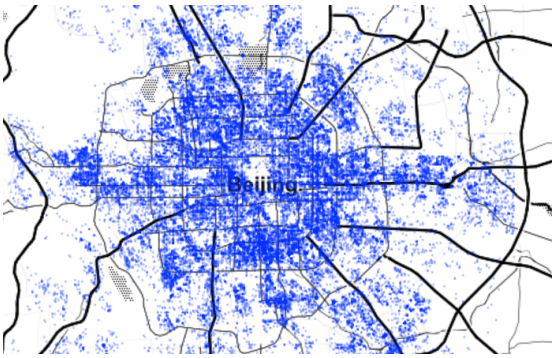

Fig. 1. A sample for the starting locations of different journeys for Mobike's free-floating system in Beijing.

A. Spatial modeling

We first analyze the data employed for the mathematical models presented in the next section. First, we overlay a grid over the service area under study, as shown in Fig. 3. The

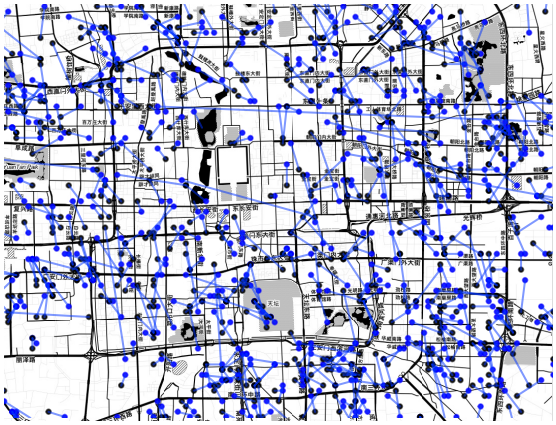

Fig. 2. A sample for the trips taken by the users in a close-up image of the service area.

grid consists of 76×109 cells, where each cell is 500 m × 500 m. The average daily demand for cell i, D_i, is shown in Fig. 4. The arrival time of each journey is not provided in the data, therefore we estimate the arrival time by calculating the journey time between the start and end locations. The distance between a start and end location is calculated using the Haversine formula, and the journey time is then estimated by assuming an average speed of 2.5 m/s.

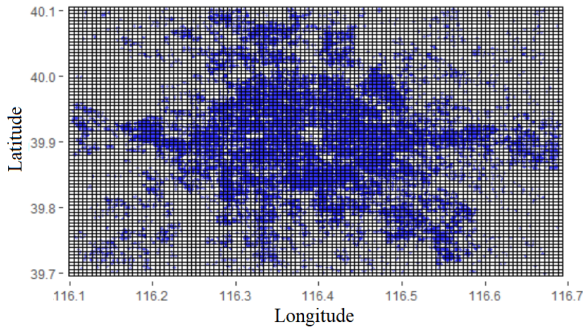

Fig. 3. Gridded map for Mobike's free-floating system with sample journeys, consisting of 76×109 cells, where each cell is 500 m × 500 m.

Fig. 4. Average daily departures for each cell. Colorcode represents the daily average number of bikes departing from each grid cell.

B. Points of interests

In addition to Mobike's data, we collected data for points of interest (POIs) located in Beijing. The locations of office buildings, metro stations, parking lots, and universities/colleges are collected from [20] and Google Places API. In total, 5,561 POIs are collected. POIs represent potential destinations for the users, with parking lots being convenient locations for parking hubs. For each type of location l (e.g., office building), we calculate the number of points of interest (L_{il}) in each cell i. For example, if there are 4 office buildings in cell 1, then $L_{1,\text{"Office Building"}} = 4$. This information is used in Section III to ensure the presence of parking hubs near POIs at locations with large demand for bikes (see (13))

C. Parking demand estimation

To calculate the parking demand in each location for a specific time instance, we determine the number of bikes present in that location. The data provides departure and arrival locations for each trip. Once a bike arrives at a location, the bike's next journey should start from the same location. However, in Mobike's data, the starting location of a journey may differ from the end location of the previous journey. This indicates that Mobike rebalances the system by manually moving some bikes. Hence, it is not possible to determine the exact number of bikes present in a specific area from the provided data. Therefore, we estimate the parking demands f_{it} at time t for $i \in \bar{G}$, where $\bar{G}$ is the set of grid cells, by simulating a rebalancing problem using linear programming. The objective of this linear program is to minimize the number of moved bikes while ensuring that the parking demand is equal to or greater than zero. The rebalancing problem is modeled as follows:

$$\min_{\{f_{it}, r_{it}, r^+_{it}\}, i\in\bar{G}, t\in T_d} \quad \sum_{i\in\bar{G}}\sum_{t\in T_d} r^+_{it} \tag{1}$$

$$\text{subject to:} \qquad r^+_{it} \geq r_{it}, i \in \bar{G}, t \in T_d, \tag{2}$$

$$\sum_{i\in\bar{G}} r_{it} = 0, t \in T_d, \tag{3}$$

$$r_{it} = 0, i \in \bar{G}, t \notin T_R, \tag{4}$$

$$f_{i,t+1} = f_{it} + A_{it} - P_{it} + r_{it}, i \in \bar{G}, t \in T_d, \tag{5}$$

$$f_{i1} = F^0_i, i \in \bar{G}, \tag{6}$$

$$\sum_{i\in\bar{G}} f_{it} \leq B, t \in T_d, \tag{7}$$

$$0.7\ F^0_i \leq f_{i|T_d|} \leq 1.3\ F^0_i, i \in \bar{G}, \tag{8}$$

$$0 \leq r^+_{it} \leq 10, f_{it} \geq 0, i \in \bar{G}, t \in T_d. \tag{9}$$

The optimization model is solved for each day separately, where T_d denotes the number of time steps per day, while we consider 5-minute time steps. The objective in (1) is to minimize the number of bikes that are moved. r_{it} takes a positive value if a bike is moved to grid cell i and a negative value if a bike is moved out of grid cell i, therefore, the summation of r_{it} over all $i \in \bar{G}$ should be zero (see (3)). r^+_{it} considers only the positive values of r_{it}, i.e., max(r_{it},0), while this is enforced by (2). Normally, the rebalancing process

occurs at specific hours in the day [4]. Constraint (4) prohibits rebalancing outside of the specified working hours, where T_R is the set of time steps between 8 A.M. - 10 P.M. as also considered in [4]. The parking demand or the fill level is defined in (5). In particular, the parking demand $f_{i,t+1}$ at $t+1$ is the sum of parking demand in the previous time instance f_{it}, arrived bikes A_{it} and the rebalancing term r_{it}, minus the departed bikes P_{it}. The initial fill level of each grid cell is defined by (6), where the value of F_i^0 is determined by identifying the first journey for each bike in the data. Constraint (7) limits the total number of bikes to the number of bikes in the system, denoted by B. We impose that the number of bikes in each cell at the end of the day should be within 30% of the starting number in (8). Finally, (9) defines the variables f_{it} and r_{it}^+, where we assume that $|r|$ must be less than 10 bikes at each time step.

The model is solved using AMPL [21] with GUROBI [22]. Fig. 5 shows a sample of the parking demand f_{it} at three randomly selected locations (grid cells) for two days. The values of f_{it} are then used in Section VI to generate parking demand scenarios, where we randomly sample different time points from f_{it} for each grid cell. The average parking demand for each cell is shown in Fig. 6.

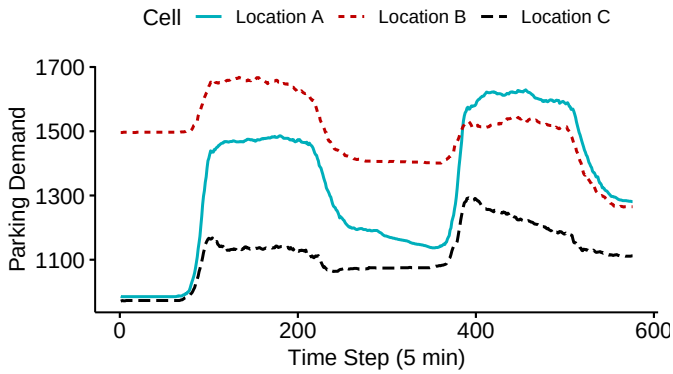

Fig. 5. Two day parking demand for three different grid cells.

Fig. 6. Average parking demand for each cell. Colorcode represents the number of parked bikes.

III. Parking Hub Optimization

One of the main concerns for micromobility service providers is where bikes or scooters are parked when they are not in use, as well as avoiding randomly parked scooters and street clutter. Fig. 7 shows the starting points of journeys that occurred in a single day between 10:00 AM and 11:00 AM, where we notice the random distribution of bikes and disorganized parking. A solution is to designate specific parking hubs (painted parking areas) for the bikes, and consumers are either not allowed to end a journey outside of a parking hub or penalized for parking outside the designated areas [3]. In order to evaluate whether the bikes are properly parked, the service provider can request photos of the parked bikes from the users through their mobile application, in addition to monitoring GPS data. In this paper, we develop methods for selecting the location of parking hubs and the number of parking spots.

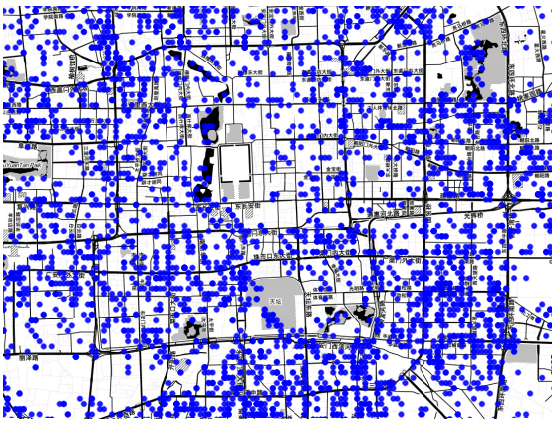

Fig. 7. Starting points of trips that occurred in a single day between 10:00 AM and 11:00 AM.

A challenge in determining the locations of parking hubs is the presence of uncertainty in parking demand. We thus denote the parking hub demand at grid cell i by δ_i, and consider three different cases: i) A deterministic set-up where δ_i is assumed to be known; ii) A worst-case set-up where δ_i is assumed to take all possible values within an interval Δ_i, and we seek to robustify our allocation with respect to that set; iii) A data-driven approach, where we consider a finite set of scenarios $\delta_{i,s}$, where $s \in S$, that the uncertain demand may take. We then enforce constraints only on those scenarios, giving rise to a scenario program.

A. Deterministic approach

We consider first the deterministic case. Each cell in Fig. 3 is considered to be a potential location for a parking hub. Since the area under study contains multiple grid cells where the demand for bikes is 0, as shown in Fig. 4, we filter out grid cells with average departure rate less than 1. The adjusted set can then be defined as $G = \{i | i \in \bar{G} \text{ and } D_i \geq 1\}$, while cell i and adjacent cells to each cell i are defined in $G(i)$. Also, we assume that only one parking hub can be placed in each cell with variable capacity u_i. The binary variable x_i is used to determine whether cell i has a parking hub, and y_{ij} is the number of bikes to be moved from cell i to j. The number of bikes that will stay at cell i is given by y_{ii}. Note that the actual number of parking hubs in a grid cell may be more than one, depending on the topographical constraints within the area. To determine the exact number of parking hubs, the selected grid cell i must be studied by using geographic information system (GIS) software to allocate the parking spots. The study in [23]

showed that customers are not willing to walk over 1000 m. Since each grid cell is 500 m × 500 m in this study, we assume bikes can only be moved to an adjacent cell; i. e., users are willing to walk to adjacent locations to park their bikes. Let $\boldsymbol{x}, \boldsymbol{u}$, and $\boldsymbol{y}$ be stacked vectors containing variables x_i, u_i, and y_{ij}, where $j \in G(i), i \in G$. We aim at determining $\boldsymbol{x}, \boldsymbol{u}$, and $\boldsymbol{y}$ by means of the following deterministic optimization problem.

$$\mathcal{P}_D: \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{y}} \quad \sum_{i\in G} C^h x_i + \sum_{i\in G} C^b u_i \tag{10}$$

$$\text{subject to:} \quad \underline{C}_i x_i \le u_i \le \bar{C}_i x_i, \tag{11}$$

$$\sum_{i\in G} u_i \ge B, \tag{12}$$

$$x_i = 1, i \in I^G, \tag{13}$$

$$\sum_{j\in G(i)} x_j \ge 1, i \in G, \tag{14}$$

$$\sum_{j\in G(i)} y_{ij} \ge \delta_i, i \in G, \tag{15}$$

$$\sum_{j\in G(i)} y_{ji} \le u_i, i \in G, \tag{16}$$

$$\sum_{i\in G} \sum_{\substack{j\in G(i)\\ i\neq j}} y_{ij} \le Y, \tag{17}$$

$$x_i \in \{0,1\}, y_{ij} \ge 0, j \in G(i), i \in G. \tag{18}$$

The objective of the optimization model comprises two terms: the cost of establishing a parking hub and the cost of each parking space. The cost coefficient C^h is a fixed cost of \$50/parking hub for establishing a parking hub, and the cost coefficient C^b is assumed to be \$4/parking space. Limits on the capacity are imposed in (11), with a lower limit $\underline{C}_i$ of 5 and an upper limit $\bar{C}_i$ of 400 parking spaces [14] (the parking spaces can be distributed in the selected area/cell). Also, the total number of parking spaces should be greater than or equal to the total number of bikes B, which is imposed by (12). Constraint (13) states that a parking hub must be installed for grid cells with POIs and high daily departure rate. The set of grid cells with POIs and sufficiently large daily departures is defined by $I^G = \{i \in G | \sum_{l\in I} L_{il} \ge 1 \text{ and } D_i \ge 30\}$, where 30 is the 0.75 quantile of the average daily departures for the dataset under study. Constraint (14) enforces that each grid cell in G should be connected to at least one parking hub. Constraint (15) states that all parking demand must be met, however, it must be within the capacity of the assigned parking hub as defined in (16). Note that since we are solving a minimization problem, constraint (15) is binding. The number of bikes to be transferred from their preferred location is limited to Y (see (17)), where we have considered $Y = 0.1B$. Finally, (18) defines the binary variable x_i and the continuous variable y_{ij}.

B. Worst-case approach

Let Δ_i be a set that contains all possible realizations, such that $\delta_i \in \Delta_i$, for all $i \in G$. The robust counterpart of $\mathcal{P}_D$ can then be formulated as follows:

$$\mathcal{P}_R: \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{y}} \max_{\{\delta_i\in\Delta_i\}_{i\in G}} \quad \sum_{i\in G} C^h x_i + \sum_{i\in G} C^b u_i \tag{19}$$
$$\text{subject to:} \quad \text{(11)-(18).}$$

Assume that for all $i \in G$, Δ_i is an interval $\Delta_i = [\underline{\delta}_i, \bar{\delta}_i]$, thus giving rise to "box" uncertainty. We determine $\underline{\delta}_i$ and $\bar{\delta}_i$ from the minimum and maximum value the demand takes in the available dataset. The uncertain parameter is only present in constraint (15), linking δ_i with y_{ij}. Since we are solving a minimization problem, the worst-case occurs when $\delta_i = \bar{\delta}_i$. $\mathcal{P}_R$ then takes the following form:

$$\mathcal{P}_R: \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{y}} \quad \sum_{i\in G} C^h x_i + \sum_{i\in G} C^b u_i$$
$$\text{subject to:} \quad \sum_{j\in G(i)} y_{ij} = \bar{\delta}_i, \forall i \in G, \tag{20}$$
$$\text{(11)-(14), (16)-(18).}$$

Problem $\mathcal{P}_R$ is a deterministic integer linear program, which can be solved using standard algorithms (e.g., branch and bound [24]). Note that (20) is enforced with equality and not inequality as in (15), as in any case this constraint would be binding. The "box" uncertainty set contains the full range of realizations for the parking demand as observed in the data. Therefore, it offers a robust choice that guarantees feasibility for the worst case scenario in the data. However, the solution can be too conservative as the probability with which uncertainty takes a specific value is not taken into account.

C. Data-driven approach

To mitigate the conservatism associated with a robust approach, we next consider a data-driven paradigm by adopting the so called scenario approach [25]. The scenario approach involves enforcing constraints on a finite number of uncertainty samples/scenarios that could be available in the form of a dataset. The optimal decision of the associated scenario program is then robust with respect to these scenarios, however, it is inherently a random quantity as repeating the problem with the same number but different scenarios would yield a different solution. The theoretical analysis of [25] allows to accompany the resulting solution with a probabilistic generalization certificate on its feasibility properties, i.e., how likely it is that the resulting solution remains feasible when it comes to a new realization of the uncertainty that is not included in the scenarios used for optimization. An appealing feature of this approach is the fact that there is no need for prior knowledge of the underlying probability distribution of the uncertainty, nor a necessity to assign a probability of occurrence to each scenario (thus implicitly assuming a discrete distribution); on the contrary the resulting certificate is distribution free and the only requirement is that scenarios are independent and identically distributed. Therefore, the scenario approach adopted here is not directly related to scenario based stochastic programming (SP). The latter would involve optimizing the expected value of a given objective criterion, and would require knowledge of at least an empirical frequency/probability per scenario. Moreover, SP is not

concerned with the generalization properties of the resulting solution when it comes to new realizations of the uncertain parameters.

Define S as the index set of independent and identically distributed (i.i.d.) extractions/scenarios of the uncertain parameter, and denote the corresponding scenarios for each cell i, by $\delta_{i,s}, s \in S$. We can then formulate the scenario program as follows:

$$\mathcal{P}_S : f(\boldsymbol{x}, \boldsymbol{u}, S) = \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{y}_s} \quad \sum_{i\in G} C^h x_i + \sum_{i\in G} C^b u_i \tag{21}$$

subject to: (11)-(14),

$$\sum_{j\in G(i)} y_{ij,s} = \delta_{i,s}, i \in G, s \in S, \tag{22}$$

$$\sum_{j\in G(i)} y_{ji,s} \leq u_i, i \in G, s \in S, \tag{23}$$

$$\sum_{i\in G} \sum_{\substack{j\in G(i) \\ i\neq j}} y_{ij,s} \leq Y, s \in S, \tag{24}$$

$$x_i \in \{0,1\}, y_{ij,s} \geq 0, j \in G(i), i \in G, s \in S. \tag{25}$$

The objective value and the deterministic constraints (11)-(14) remain the same. Constraints (15)-(17) are scenario dependent, therefore, they are converted to (22)-(24). Notice that the subscript s is now introduced to $y_{ij,s}$ to indicate that these are scenario dependent variables, and hence different for each scenario $s \in S$. The resulting scenario program leads to a less conservative solution, however, the size of the associated problem grows with scenarios, thus becoming prohibitive to solve for a realistic number of scenarios. To alleviate the latter, we propose a scenario decomposition method to solve $\mathcal{P}_S$ in the next section.

IV. Decomposition Methodology

To solve large scale instances of the data-driven parking hub optimization problem in $\mathcal{P}_S$, we rely on decomposition techniques based on ADMM. ADMM solves optimization problems by decomposing them into smaller subproblems, combining the principles of dual decomposition and augmented Lagrangian. Many variants have been developed for ADMM, we focus here on Consensus ADMM, and refer to it hereinafter as ADMM. For detailed information on ADMM and its convergence, the reader is referred to [17].

Since convergence results related to ADMM rely on the underlying optimization problem being convex, prior to solving $\mathcal{P}_S$, we determine the locations of the parking hubs, thus fixing the binary variable x_i for each cell. To achieve this, we first solve the worst case problem in $\mathcal{P}_R$ and denote the returned solution by x_i^*. $\mathcal{P}_R$ has fewer constraints compared to $\mathcal{P}_S$, as the worst case problem does not grow with the number of scenarios. Once these variables are fixed, ADMM is then used to determine the number of parking spaces. Therefore, ADMM determines a suboptimal solution for the parking hub allocation problem.

A. Consensus ADMM

The idea of ADMM is to decompose the presented problem into several subproblems, and then coordinate the solutions by solving each subproblem separately. $\mathcal{P}_S$ is decomposed across the scenarios, where the common variable is u_i, and x_i is fixed. We duplicate the variable u_i, so that each scenario will have its own local capacity variable u_{is}. Let z_i be the global variable representing the capacities of the cells, while we enforce the so called consistency constraints $z_i - u_{is} = 0$. We can then define λ_{is} as the dual variable associated with the constraint $z_i - u_{is} = 0$. Using ρ as a penalty parameter (step size), we can denote the augmented Lagrangian associated with $\mathcal{P}_S$ as:

$$\begin{aligned} L_\rho(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{s\in S} \big(\sum_{i\in G} C^h x_i^* + \sum_{i\in G} C^b u_{is} \\ + \sum_{i\in G} \lambda_{is}(z_i - u_{is}) + \frac{\rho}{2} \sum_{i\in G} (z_i - u_{is})^2 \big). \end{aligned} \tag{26}$$

At each iteration k, the ADMM algorithm consists of three steps, update u_{is}, z_i, then λ_{is}. The pseudocode for ADMM is given in Algorithm 1.

Algorithm 1 starts by sampling the scenarios. In step 2, we initialize the variables and parameters. The variable x_i is then found in step 3 by solving $\mathcal{P}_R$. The ADMM algorithm starts at step 4. In step 5, the minimization is performed with respect to $\boldsymbol{u}_s = \{u_{1s}, ..., u_{|G|s}\}$ and $\boldsymbol{y}_s$, where $\mathcal{C}_s$ encodes the set of constraints for scenario s coupling these variables; i.e., (11)-(14) and (22)-(25). This step is separable across scenarios, with z_i and λ_{is} fixed to their values at iteration k of the algorithm. Variable z_i is updated by solving the augmented Lagrangian as follows:

$$z_i = \arg\min_{\mathbf{z}} \sum_{s\in S} \sum_{i\in G} \big(\lambda_{is} z_i + \frac{\rho}{2} (z_i - u_{is})^2 \big). \tag{27}$$

This is an unconstrained quadratic minimization problem which can be performed analytically, yielding the update in (29). The dual variable λ_{is} is updated by means of a gradient ascent update in step 7. The steps are repeated until a stopping criterion is reached. In this study we use a combination of three different criteria: 1) Time limit T; 2) Iteration limit K; and 3) Stopping tolerance ϵ^{tol} on the residual error; i.e., $||z_i - u_{is}||_\infty \leq \epsilon^{\text{tol}}$. Upon convergence, (31) takes the maximum value between the global and local variables to ensure feasibility. The variable is also rounded up to the nearest integer ($\lceil \cdot \rceil$ denotes the nearest integer greater than its argument), as parking spaces should be integer. We denote the resulting solution by $\boldsymbol{u}^*$, and by $\boldsymbol{y}^*$ the associated scenario dependent decision vector returned by the ADMM algorithm.

Despite being convergent, the convergence rate of ADMM might be slow for a large number of scenarios. To speed-up convergence, we propose in the next subsection a modification of the ADMM algorithm, combining it with principles from Constraint-and-Column Generation.

B. CC-ADMM

In scenario optimization problems, it is often not necessary to include all scenarios to obtain the optimal solution. Let S be the sampled scenarios and $\mathcal{N} \subset S$. If we solve

Algorithm 1 ADMM

1: Sample S scenarios from the dataset
2: Initialize $k, u_{is}, z_i, \lambda_{is}, \rho$
3: Solve $\mathcal{P}_R$ to find x_i^*
4: **while** $k \leq K$ and time elapsed $\leq T$ **do**
5: For $s \in S$, update u_{is} by

$$u_{is} = \underset{(\mathbf{u}_s, \mathbf{y}_s) \in \mathcal{C}_s}{\arg\min} \sum_{i \in G} C^h x_i^* + \sum_{i \in G} C^b u_{is} - \sum_{i \in G} \lambda_{is} u_{is} + \frac{\rho}{2} \sum_{i \in G} (z_i - u_{is})^2 \tag{28}$$

6: For $i \in G$, update z_i by

$$z_i = \frac{1}{|S|} \sum_{s \in S} u_{is} - \frac{1}{\rho |S|} \sum_{s \in S} \lambda_{is} \tag{29}$$

7: For $i \in G, s \in S$, update λ_{is} by

$$\lambda_{is} = \lambda_{is} + \rho(z_i - u_{is}) \tag{30}$$

8: **if** $||z_i - u_{is}||_\infty \leq \epsilon^{\text{tol}}$ **then**
9: **break** (ADMM converged)
10: **end if**
11: Update the iteration number $k = k + 1$
12: **end while**
13: Set the value of u_i^* using:

$$u_i^* = \lceil \max\{z_i, \max_{s \in S} u_{is}\} \rceil, i \in G \tag{31}$$

$\mathcal{P}_S$ with $\mathcal{N}$ scenarios and the resulting solution is feasible for the remaining scenarios $S \backslash \mathcal{N}$, then the solution is also optimal for S. Let $f(\boldsymbol{x}^*, \boldsymbol{u}, S)$ and $f(\boldsymbol{x}^*, \boldsymbol{u}, \mathcal{N})$ denote the objective functions for the scenario optimization problem $\mathcal{P}_S$ with scenario sets S and $\mathcal{N}$, respectively, once the binary variables are fixed. Since $\mathcal{N} \subset S$, if the optimal solution[1] $u_i^*(\mathcal{N}) = \arg\min(f(\boldsymbol{x}^*, \boldsymbol{u}, \mathcal{N}))$ is feasible for $S \backslash \mathcal{N}$, then $f(\boldsymbol{x}^*, \boldsymbol{u}^*, \mathcal{N}) = f(\boldsymbol{x}^*, \boldsymbol{u}^*, S)$ and $\mathcal{N}$ is called a support scenario set, as solving the problem only with these scenarios leads to the same solution with the one that would have been obtained if all scenarios were employed. Notice that $u_i^*(\mathcal{N})$ has $\mathcal{N}$ as an argument to emphasize the dependency on particular set of scenarios.

CC-ADMM uses a cutting plane procedure with the aim of determining a support scenario set $\mathcal{N}$. In the proposed CC-ADMM algorithm, instead of solving ADMM for all scenarios, we iteratively increase the number of scenarios and solve the problem using ADMM. The obtained solution is then checked against the remaining scenarios at each iteration of the algorithm. CC-ADMM contains a master and a slave problem. The master problem solves $\mathcal{P}_S$, using $\mathcal{N}$ rather than S scenarios using ADMM, while the slave problem checks feasibility of the resulting solution with respect to constraints (22)-(24) for each scenario separately. The slave problem generates new constraints in (22)-(24) and variables/columns ($y_{ij,s}$) for the master problem if the slave problem is infeasible. The CC-ADMM algorithm is summarized in Algorithm 2.

[1]We assume for simplicity that for any set of scenarios S, the optimal solution of $\mathcal{P}_S$ is unique. In the opposite case a tie-break rule can be employed to single-out one out of the possibly multiple minimizers of $\mathcal{P}_S$.

Algorithm 2 CC-ADMM

1: Sample S scenarios
2: Initialize $k, u_{is}, z_i, \lambda_{is}, \rho$
3: Solve $\mathcal{P}_R$ to find x_i^*
4: $\mathcal{N} \leftarrow$ sample a single scenario from S
5: Set flag $= 0$
6: **while** flag $= 0$ and time elapsed $\leq T$ **do**
7: **while** $k \leq K$ **do** (Master Problem)
8: For $s \in \mathcal{N}$, update u_{is} using (28)
9: For $i \in G$, update z_i by

$$z_i = \frac{1}{|\mathcal{N}|} \sum_{s \in \mathcal{N}} u_{is} - \frac{1}{\rho |\mathcal{N}|} \sum_{s \in \mathcal{N}} \lambda_{is} \tag{32}$$

10: For $i \in G, s \in \mathcal{N}$, update λ_{is} using (30)
11: **if** $||z_i - u_{is}||_\infty \leq \epsilon^{\text{tol}}$ **then**
12: Set flag $= 1$ and **break** (go to step 16)
13: **end if**
14: Update the iteration number $k = k + 1$
15: **end while**
16: Set the value of u_i^* using (31)
17: Define $S' = S \backslash \mathcal{N}$
18: **while** $|S'| > 0$ **do** (Slave Problem)
19: Let $n \leftarrow$ sample a single scenario from S'
20: Let $S' \leftarrow S' \backslash \{n\}$
21: Check the feasibility of (22)-(24)

$$\min_{\boldsymbol{y}} \{0 : \text{s.t. (22)-(24)}, u_i = u_i^*, \delta_i = \delta_{i,n}, \ \forall i\} \tag{33}$$

22: **if** infeasible **then**
23: Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$ (update support set)
24: Let $k = 1$ (reset iteration number)
25: Set flag $= 0$ and **break** (return to step 7)
26: **end if**
27: **end while**
28: **end while**

Steps 1-5 in Algorithm 2 are initialization steps. The algorithm starts by sampling S scenarios in step 1. Step 2 defines the initial values of the ADMM parameters. Step 3 solves $\mathcal{P}_R$ to determine x_i^*, which is fixed for the rest of the algorithm. In step 4, we randomly select a scenario from S and assign it to $\mathcal{N}$. Step 5 initializes flag, which will indicate the end of the algorithm if flag $= 1$. Steps 6-16 solve the master problem $\mathcal{P}_S$ using ADMM, as explained in Section IV-A. The value of u_i^* is set in step 16 using (31). Steps 17-26 solve the slave problem, where for each scenario, the feasibility of $\boldsymbol{u}^*$ is checked with respect to all scenarios not included in $\mathcal{N}$. We randomly sample a scenario from $S' = S \backslash \mathcal{N}$ and solve a feasibility problem in step 21 to decide on whether $\boldsymbol{u}^*$ satisfies the constraints corresponding to the scenario indexed by n. If this problem is infeasible, we add the corresponding scenario to $\mathcal{N}$ and return to step 8. If all scenarios are feasible, then the solution is optimal and flag $= 1$, which terminates the algorithm. CC-ADMM not only improves the computational performance of ADMM, but allows determining $|\mathcal{N}|$; i.e.,

the cardinality of a support set. It is shown in Section V that this quantity is crucial in accompanying $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ with a certificate on the probability that it remains feasible when a new realization of the uncertainty is encountered.

V. Robustness and Constraint Violation

Denote by $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ the optimal solution of $\mathcal{P}_S$ as this is returned by the CC-ADMM algorithm. As mentioned in footnote 1, we assume for simplicity that $\mathcal{P}_S$ admits a unique solution for any S; our analysis remains valid even in the case where multiple solutions exist, however, it would require then to adopt a tie-break rule to select one solution among the possibly many minimizers. The optimal solution depends on the $|S|$ scenarios $\boldsymbol{\delta}^1, ..., \boldsymbol{\delta}^{|S|}$, where $\boldsymbol{\delta}^{|S|} = \delta_{1|S|}, ..., \delta_{|G||S|}$ employed, hence, it is itself a random quantity.

We aim at quantifying the probability with which this solution violates at least one of the constraints in (11)-(18) when a new realization of the uncertainty is encountered. In our context, constraint violation indicates insufficient capacity (parking spots) for the bikes. To address this problem we rely on results based on the so called scenario approach [25], and in particular build on the recent developments in [26], [27]. To this end, for any fixed $\beta \in (0, 1)$, let $\varepsilon : \ S \rightarrow [0, 1]$ be a function that satisfies

$$\begin{aligned} &\varepsilon(|S|) = 1, \ \text{and} \\ &\sum_{k=0}^{|S|-1} \binom{|S|}{k} (1 - \varepsilon(k))^{|S|-k} = \beta. \end{aligned} \tag{34}$$

We then have the following proposition, which is shown without proof as it follows by a direct application of Theorem 1 in [26], since $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ is by construction feasible for the constraints in $\mathcal{P}_S$ for all uncertainty scenarios in $\{\boldsymbol{\delta}^1, ..., \boldsymbol{\delta}^{|S|}\}$.

Proposition 1: Fix $\beta \in (0, 1)$, and consider $\epsilon(\cdot)$ satisfying (34). Let $|\mathcal{N}|$ denote the number of support scenarios retuned by Algorithm 2. We then have that

$$\begin{aligned} &\mathbb{P}^{|S|}\big\{\boldsymbol{\delta}^1, ..., \boldsymbol{\delta}^{|S|} : \mathbb{P}\{\bar{\boldsymbol{\delta}} \in \boldsymbol{\Delta} : \ \text{for all } \boldsymbol{y}, \\ &\quad (\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{y}) \text{ is not feasible for (11)-(18) with } \boldsymbol{\delta} = \bar{\boldsymbol{\delta}}\} \\ &\quad \leq \varepsilon(|\mathcal{N}|)\big\} \geq 1 - \beta. \end{aligned} \tag{35}$$

The inner probability in (35) encodes the probability of constraint violation, i.e., the probability of the event that a new realization of the parking demand $\bar{\boldsymbol{\delta}}$ is encountered and for all $\boldsymbol{y}$, $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{y})$ violates at least one constraint in (11)-(18). Proposition 1 implies then that the probability of constraint violation is at most equal to a given threshold $\varepsilon(|\mathcal{N}|)$, with confidence at least equal to $1-\beta$. Note that $\boldsymbol{y}$ is a second-stage decision vector that depends on the uncertainty realization; we have constraint satisfaction if at least one choice for $\boldsymbol{y}$ exists, such that together with $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{y})$ is feasible for (11)-(18). It should be noted that the probabilistic statement of Proposition 1 quantifies the feasibility properties of the first stage variables $\boldsymbol{x}^{**}$ and $\boldsymbol{u}^*$ that are scenario independent; this is due to the fact that second stage decisions act as certificates and a solution tuple would be considered as feasible if one such decision exists so that (11)-(18) are satisfied, without being interested in the exact value of that decision.

Typically, one sets β to very small numerical values ($1 - \beta$ becomes high), so that constraint violation lower than a given threshold occurs with very high confidence. In this case, the quality of (35) depends on $\varepsilon(|\mathcal{N}|)$. The lower $\varepsilon(|\mathcal{N}|)$, the lower the probability that at least one constraint is violated when a new $\bar{\boldsymbol{\delta}}$ is encountered. By (34), it follows that $\varepsilon(\cdot)$ is a non-decreasing function of each argument, hence the lower $|\mathcal{N}|$ the more informative the result of Proposition 1 becomes. This highlights the importance of the cardinality of the support scenario set in accompanying the resulting solution with probabilistic feasibility certificates.

Calculating the cardinality of the support set (without resorting to conservative upper bounds) is in general difficult. In [26] a greedy algorithm is proposed which is, however, sensitive to numerical errors and in our context would require solving the iterative algorithm selected to solve $\mathcal{P}_S$ a total of $|S|$ times. However, CC-ADMM offers directly an estimate of $\varepsilon(|\mathcal{N}|)$, preventing the excessive computational burden of employing the greedy algorithm. Moreover, note that if the problem under study is non-degenerate (see [27] for a definition), a condition which is, however, hard to verify, (34) could be directly replaced by another expression which leads to tighter values of $\varepsilon(|\mathcal{N}|)$ (see Theorem 2 in [27]).

By splitting β equally to the $|S|$ terms for the summation in (34), we can obtain an explicit expression for $\varepsilon(k)$, which is sufficient for the satisfaction of (34). This is given by

$$\varepsilon(k) = \begin{cases} 1, & k = |S| \\ 1 - \sqrt[|S|-k]{\dfrac{\beta}{|S|\binom{|S|}{k}}}, & k < |S|. \end{cases} \tag{36}$$

It is apparent from (36) that the smaller $\mathcal{N}$, the lower $\varepsilon(|\mathcal{N}|)$ becomes.

VI. Simulation Results

We illustrate the developed methodologies to allocate parking hubs in Beijing for Mobike's FFBS. The parking demand f_{it}, for each grid cell i and for all time instances t, as estimated in Section II-C, is randomly sampled to generate the parking demand scenarios. The available data allow for 4032 scenarios; we denote this set of scenarios by $\bar{S}$, with $|\bar{S}|$ = 4032. Denote by $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ the optimal solution of the scenario program $\mathcal{P}_S$, as this is returned by the methodologies outlined in the previous section, when a scenario set $S \subset \bar{S}$ is employed. We calculate the out-of-sample violation probability of $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ by checking its feasibility with respect to scenarios that are not included in S, i.e., for scenarios $s \in \bar{S} \backslash S$. The optimization problem is modelled using AMPL [21] and solved using GUROBI 9.0 [22]. The problem is solved on a PC with Intel Core i7-8550U 1.8 GHz CPU and 16 GB RAM.

A. "Box" uncertainty

We determine the bounds of the parking demand $\delta_i \in [\underline{\delta}_i, \bar{\delta}_i]$ by computing the maximum and minimum among the scenarios included in our data set, as also performed in [28]. Different sample sizes of scenarios $|S|$ are selected, ranging from 200 to 2000. The simulation results are shown in Table I. Solutions for the different samples are all obtained

TABLE I
PARKING ALLOCATION RESULTS USING "BOX" UNCERTAINTY

$\|S\|$	Objective Value	Hubs	Parking Spots	Out-of-Sample Violation[1]
2000	$2,575,238	5163	579,272	0%
1000	$2,571,034	5149	578,396	0%
800	$2,568,464	5130	577,991	0%
600	$2,565,030	5113	577,345	0%
400	$2,556,034	5067	575,671	0.14%
200	$2,539,290	4981	572,560	0.40%

[1] The scenarios in the out-of-sample test are given by $\bar{S}\backslash S$.

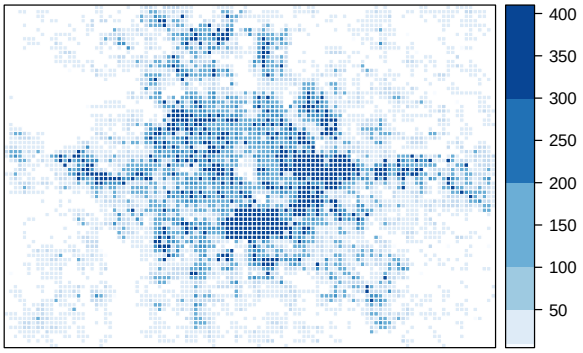

Fig. 8. Allocated parking hubs using robust optimization with "box" uncertainty and 2000 uncertainty scenarios. Colorcode represents the number of parking spots.

TABLE II
PARKING ALLOCATION RESULTS USING CC-ADMM

$\|S\|$	$\|\mathcal{N}\|$	Objective Value	Parking Spots	Out-of-sample Violation[1]	$\varepsilon(\|\mathcal{N}\|)$	Travel[2] (hrs)
2000	31	$2,158,190	475,010	0.00%	8.7%	0
1000	29	$2,157,166	474,929	0.24%	14.3%	3
800	26	$2,159,788	475,822	0.84%	15.7%	11
600	26	$2,160,058	476,102	0.90%	19.5%	12
400	19	$2,159,118	476,442	1.38%	21.8%	19
200	21	$2,156,798	476,937	2.51%	37.4%	31

[1] The scenarios in the out-of-sample test are given by $\bar{S}\backslash S$.
[2] The estimated total extra travel time for the consumers due to unavailable parking spaces.

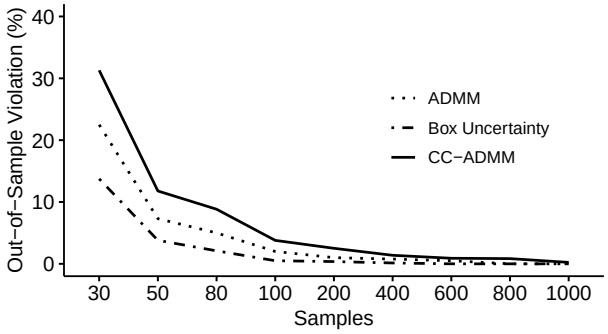

Fig. 9. Percentage of out-of-sample probability of constraint violation for ADMM, "box" uncertainty, and CC-ADMM with $|S| = 30, \ldots, 1000$.

in less than 2 minutes. The objective value represents the total cost (in dollars) of establishing the parking spots. Fig. 8 shows the locations and sizes of the selected parking hubs. The results of the "box" uncertainty methodology to solve $\mathcal{P}_R$ leads to a conservative performance. The out-of-sample violation probability is low even with 200 scenarios, while the parking spots in Table I are more than 22% higher than the total number of bikes. In the next subsection, employing CC-ADMM to obtain a solution of the data-driven problem $\mathcal{P}_S$ shows that the number of parking spots, i.e., the capacity of the parking hubs as dictated by $\boldsymbol{u}^*$, do not have to be much higher than the number of bikes to achieve a less conservative solution with higher but moderate out-of-sample probability of constraint violation.

B. Data-driven approach

We solve the scenario problem $\mathcal{P}_S$ using ADMM and CC-ADMM, with design parameters set to $\rho = 10$ and $\epsilon^{\text{tol}} = 0.5$. We impose an 8-hour time limit, and the maximum number of iterations is 100. The results for CC-ADMM with 200 to 2,000 scenarios are shown in Table II. The number of parking hubs are the same as Table I. To obtain the optimal solution, CC-ADMM did not require more than 31 support scenarios in the presented cases, and the highest computation time is 194 min. Compared to "box" uncertainty, the objective value is improved by approximately 16%, while the out-of-sample probability of violation is less than 1% when the scenarios employed for optimization purposes ($|S|$) are more than 400. The last column in Table II shows the potential travel time that the consumers will incur due to parking space unavailability. We estimate this value by calculating the distance between the desired destination and the closest location with an open parking space in the violated scenarios, while assuming the walking speed is 5 km/hr. Notice that even though the number of parking spots is higher with $|S| \leq 600$, the additional travel times are higher than the other solutions, which indicates a poor distribution of parking spaces.

The out-of-sample probability of constraint violation for the three methods of Section IV are compared in Fig. 9. As expected, the worst-case approach based on "box" uncertainty is more conservative and as a result exhibits a lower violations. However, for $|S| \geq 400$, the out-of-sample violations are mostly comparable among the three methods. Using (34) with $\beta = 10^{-6}$, we calculate an *a posteriori* bound $\epsilon(|\mathcal{N}|)$ on the probability of constraint violation, which can be interpreted as the probability of overcapacity (or lack of parking spot). When $|S|$ = 2000, $\epsilon(|\mathcal{N}|) = 8.7\%$ (see also second last column in Table II). The obtained theoretical bounds for different values of $|S|$ are also contrasted with the out-of-sample ones in Fig. 10. When the number of scenarios employed for optimization purposes is less than 600, the theoretical bound is high, exceeding 20% thus rendering the result of Proposition 1 not of practical use for such cases; however, this becomes relevant once the number of scenarios increases. Fig. 11 illustrates the allocated parking hubs using CC-ADMM for the case where 2000 scenarios are employed.

Fig. 12 compares the objective value of $\mathcal{P}_R$ using "box" uncertainty, with the one of $\mathcal{P}_S$ as returned by the ADMM and the CC-ADMM algorithm. It can be observed that robust optimization with "box" uncertainty is more conservative, leading consistently to a higher objective value compared to the one achieved by ADMM and the CC-ADMM for

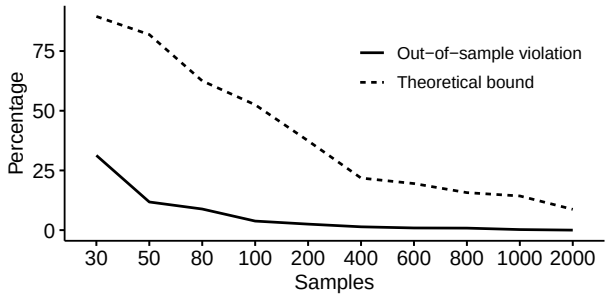

Fig. 10. Percentage of out-of-sample probability of constraint violation and the theoretical bound as obtained from Proposition 1.

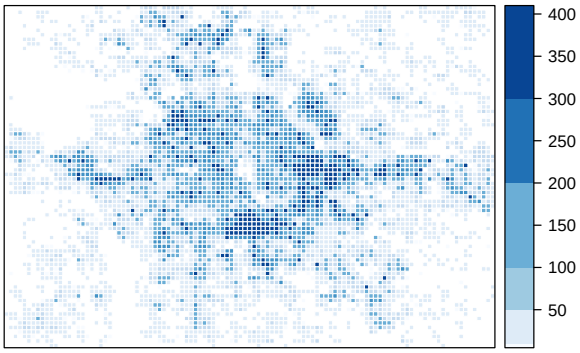

Fig. 11. Allocated parking hubs using CC-ADMM and 2000 uncertainty scenarios. Colorcode represents the number of parking spots.

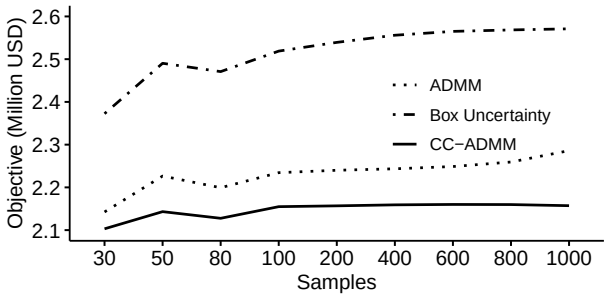

Fig. 12. The optimal objective value achieved by ADMM and CC-ADMM to solve $\mathcal{P}_S$, and by the solution of $\mathcal{P}_R$ when "box" uncertainty is considered, with respect to the number of samples $|S|$ = 30,...,1000.

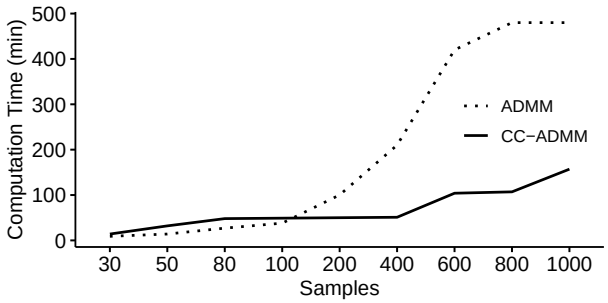

Fig. 13. A comparison between the computation time of ADMM and CC-ADMM with respect to the number of samples $|S|$ = 30,...,1000.

the data driven problem. Note that CC-ADMM and ADMM converge to the same solution since they solve the same problem (26). However, this is not apparent from Fig. 12, as the solution of each algorithm depends on the stopping criteria. When a stopping criterion is reached, the algorithms terminate, at which point the solution obtained might be different among the two algorithmic alternatives. In terms of computation times, we compare CC-ADMM and ADMM in Fig. 13. For CC-ADMM, the GUROBI solver is used in step 3 and within each iteration for steps 8 and 21, where the average computation times are 64 s, 0.86 s, and 0.47 s, respectively. The computation times shown in Fig. 13 are the accumulation of the GUROBI runs. When the number of scenarios is low, ADMM is faster than CC-ADMM, since the latter iteratively increases the number of scenarios to approach the optimal solution. On the other hand, for a large number of scenarios, the imposed time limit stopping criterion is reached prior to ADMM convergence, while CC-ADMM converges in less than 4 hours. This behaviour stems from the fact that, in contrast to ADMM, CC-ADMM only requires considering up to 31 scenarios (see $\mathcal{N}$ column in Table II). The reported computation times for CC-ADMM and ADMM can be further improved by solving (28) in parallel across different processors.

Next, we perform a sensitivity analysis on CC-ADMM over the value of ρ. The results of six CC-ADMM simulations with different values of ρ are shown in Table III. Increaseing the parameter leads to a decrease in computation time. However, notice that the objective value decreases when ρ is increased, and then increases again with larger values of ρ. Therefore, we must select a value that balances the computation time and quality of the solution. With small values of ρ, the computation time is high and the master problem (steps 7-15 in Algorithm 2) may not converge, which can lead to conservative values for u_i. For large values of ρ, the algorithm converges too fast to a conservative solution. The empirical results indicate that a value around $\rho = 10$ achieves a good tradeoff between solution time and quality.

TABLE III
ALLOCATION RESULTS USING CC-ADMM WITH DIFFERENT VALUES OF ρ

ρ	Comp. Time (min)	Objective	$\lvert\mathcal{N}\rvert$
0.1	336	\$2,188,440	23
1	312	\$2,163,730	25
5	267	\$2,159,510	26
10	157	\$2,157,166	29
100	105	\$2,251,230	22
1000	43	\$2,323,370	19

In summary, the solution of the data driven problem $\mathcal{P}_S$ returned by the CC-ADMM algorithm, is less conservative compared to the one of $\mathcal{P}_R$ using "box" uncertainty, thus leading to a lower objective value. It is worth mentioning that even when only 30 scenarios are used to formulate the uncertainty "box", the resulting cost is still significantly higher compared to the one achieved by CC-ADMM using 2000 uncertainty scenarios. Moreover, we can accompany the solution returned by CC-ADMM with probabilistic feasibility certificates; using 2000 scenarios the probability of constraint violation for a new unseen uncertainty realization is, with high confidence, at most 9%. At the same time, the computational time requirements of CC-ADMM are much lower compared

to ADMM, thus highlighting its appealing features. Note that solving the scenario program $\mathcal{P}^S$ directly using GUROBI does not yield a solution due to the scale of the system and the complicated nature of the facility location problem, even with a small number of scenarios.

VII. Conclusion

We proposed a scenario optimization method for solving the parking allocation problem in free-floating bike-share systems. We modelled the problem using integer programming with uncertain parking demand, and designed an algorithm that combines Constraint-and-Column Generation with ADMM. The algorithm determines a subset of scenarios that is sufficient for achieving the same optimal value with the one that would be obtained if all scenarios were utilized. Our method allows solving large-scale scenario optimization problems with many uncertain parameters using a moderate number of scenarios, while also providing probabilistic guarantees on the probability that the resulting solution violates a new, yet unseen, realization of the uncertainty. Numerical results showed that the proposed method outperforms robust optimization with box uncertainty in terms of conservatism of the solution, and the classical ADMM method in terms of computational time.

Acknowledgment

The authors would like to thank Agne Milukaite and Peter Ebsen (Cycle.land, Oxford) for useful discussions on the topic that motivated this work.

References

[1] J. A. Hirsch, J. Stratton-Rayner, M. Winters, J. Stehlin, K. Hosford, and S. J. Mooney, "Roadmap for free-floating bikeshare research and practice in North America," *Transp. Rev.*, vol. 39, no. 2, pp. 706-732, Aug. 2019.

[2] Nice Ride Minnesota, "Nice Ride dockless FAQs," [Online]. Available: https://www.niceridemn.com/how-it-works/meet-the-bikes/dockless/dockless-faqs. Accessed on: January 3, 2019.

[3] Beryl Bikes, "How Beryl Bikes work," [Online]. Available: https://beryl.cc/bikeshare/bournemouth-poole. Accessed on: January 3, 2019.

[4] J. Pfrommer, J. Warrington, G. Schilbach, M. Morari, "Dynamic vehicle redistribution and online price incentives in shared mobility systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1567-1578, Aug. 2014.

[5] Z. Yang, J. Chen, J. Hu, Y. Shu and P. Cheng, "Mobility modeling and data-driven closed-loop prediction in bike-sharing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4488-4499, Dec. 2019.

[6] C. Park, S. Y. Sohn, "An optimization approach for the placement of bicycle-sharing stations to reduce short car trips: An application to the city of Seoul," *Transp. Res. Part A: Policy and Practice*, vol. 105, no. 1, pp. 154-166, Nov. 2017.

[7] Y. Ren, *et al.*, "Rebalancing bike sharing systems for minimizing depot inventory and traveling costs," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3871-3882, Sep. 2020.

[8] M. Yuan, Q. Zhang, B. Wang, Y. Liang, H. Zhang, "A mixed integer linear programming model for optimal planning of bicycle sharing systems: A case study in Beijing," *Transp. Res. Part A: Policy and Practice*, vol. 47, Mar. 2019.

[9] Y. Ji, X. Jin, X. Ma and S. Zhang, "How does dockless bike-sharing system behave by incentivizing users to participate in rebalancing?," *IEEE Access*, vol. 8, pp. 58889-58897, Mar. 2020.

[10] G. Cheng, Y. Guo, Y. Chen and Y. Qin, "Designating city-wide collaborative geofence sites for renting and returning dock-less shared bikes," *IEEE Access*, vol. 7, pp. 35596-35605, Mar. 2019.

[11] T. Shen, K. Hua and J. Liu, "Optimized public parking location modelling for green intelligent transportation system using genetic algorithms," *IEEE Access*, vol. 7, pp. 176870-176883, Dec. 2019.

[12] H. Park, A. Shafahi, A. Haghani, "A stochastic emergency response location model considering secondary incidents on freeways," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2528-2540, Mar. 2016.

[13] Z. Sun, Y. Li, and Y. Zuo, "Optimizing the location of virtual stations in free-floating bike-sharing systems with the user demand during morning and evening rush hours," *J. Adv. Transp.*, vol. 2019, pp. 1-11, Mar. 2019.

[14] M. Hua, X. Chen, S. Zheng, L. Cheng, and J. Chen, "Estimating the parking demand of free-floating bike sharing: A journey-data-based study of Nanjing, China," *J. Cleaner Prod.*, vol. 244, no. 1, pp. 1-11, Jan. 2020.

[15] M. Cheng and W. Wei, "An AHP-DEA approach of the bike-sharing spots selection problem in the free-floating bike-sharing system," *Discrete Dyn. Nature Soc.*, vol. 2020, Sep. 2020.

[16] L. V. Snyder, "Facility location under uncertainty: A review," *IIE Trans.*, vol. 38, no. 7, pp. 547-564, Feb. 2007.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1-122, 2010.

[18] B. Zeng and L. Zhao, "Solving two-stage robust optimization problems using a column-and-constraint generation method," *Oper. Res. Lett.*, vol. 41, no. 5, pp. 457-461, Sep. 2013.

[19] Mobike, "Mobike big data challenge 2017," bien data competitions, June 25, 2017. [Online]. Available: https:// www.biendata.com/ competition/mobike/. Accessed on: December 1, 2019.

[20] X. Lu, Collect logistics data in Beijing, (2018), GitHub repository, https://github.com/xclu/logisticsBeijing. Accessed on: January 3, 2020.

[21] R. Fourer, D. M. Gay, and B. Kernighan, "AMPL: a mathematical programming language," *Algorithms and Model Formulations in Math. Programming*, S. W. Wallace, Ed. New York, NY, USA: Springer-Verlag, 1989, pp. 150-151.

[22] "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com/documentation/9.0/refman.

[23] A. Singla, M. Santoni, G. Bartok, P. Mukerji, M. Meenen, and A. Krause, "Incentivizing users for balancing bike sharing systems," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 723-729.

[24] M. A. Shafia, M. Pourseyed Aghaee, S. J. Sadjadi and A. Jamili, "Robust train timetabling problem: mathematical model and branch and bound algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 307-317, March 2012.

[25] G. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Trans. Automat. Control*, vol. 51, no. 5, pp. 742-753, May 2006.

[26] M. C. Campi, S. Garatti and F. A. Ramponi, "A general scenario theory for nonconvex optimization and decision making," *IEEE Trans. Automat. Control*, vol. 63, no. 12, pp. 4067-4078, Dec. 2018.

[27] M. C. Campi and S. Garatti, "Wait-and-judge scenario optimization," *Math. Program., Ser. B*, pp. 1-35, Jul. 2016.

[28] K. Margellos, P. J. Goulart, and J. Lygeros, "On the road between robust optimization and the scenario approach for chance constrained optimization problems," *IEEE Trans. Automat. Control*, vol. 59, no. 8, pp. 2258-2263, Aug. 2014.

Anmar Arif (M'19) is an Assistant Professor at King Saud University and a Research Associate at the University of Manchester. He received his Ph.D. in Electrical and Computer Engineering from Iowa State University, Ames, IA, USA. He received his B.S. and MSE degrees in electrical engineering from King Saud University and Arizona State University in 2012 and 2015, respectively. He was Academic Visitor at the University of Oxford from 2019 to 2021. His research interest includes power system optimization, renewable energy integration, transportation system electrification, and operations research.

Kostas Margellos (S'09-M'14) received the Diploma degree in electrical engineering from the University of Patras, Patras, Greece, in 2008, and the Ph.D. degree in control engineering from ETH Zurich, Zurich, Switzerland, in 2012. During 2013, 2014, and 2015, he was a Postdoctoral Researcher at ETH Zurich, UC Berkeley, and Politecnico di Milano, respectively. In 2016, he joined the Control Group, Department of Engineering Science, University of Oxford, where he is currently serving as an Associate Professor. He is also a Fellow in AI & Machine Learning at Reuben College, and a Lecturer at Worcester College. His research interests include optimization and control of complex uncertain systems, with applications to energy and transportation networks.