Rebalancing in Vehicle-sharing Systems with Service Availability Guarantees

Michal Čáp and Tomáš Roun

***Abstract*— A station-based vehicle sharing system consists of a fleet of vehicles (usually bikes or cars) that can be rented at one station and returned at another station. We study how to achieve guaranteed service availability in such systems. Specifically, we are interested in determining a) the fleet size and b) a vehicle rebalancing policy that guarantees that a) every customer will find an available vehicle at the origin station and b) the customer will find a free parking spot at the destination station. We model the evolution of the number of vehicles at each station as a stochastic process. The proposed rebalancing strategy iteratively solves a chance-constrained optimization problem to find a rebalancing schedule that ensures that no service failures will occur in the future with a given level of confidence. We show that such a chance-constrained optimization problem can be converted into a linear program and efficiently solved. As a case study, we apply the proposed method to control a simulated bike-sharing system in Boston using real-world historical demand. Our results demonstrate that our method can indeed ensure the desired level of service availability even when the demand does not fully conform to the assumptions of the underlying stochastic model. Moreover, compared with a state-of-the art rebalancing method, the proposed method can achieve nearly full service availability while making less than half of the rebalancing trips.**

I. INTRODUCTION

Bike sharing and car sharing systems can act as a sustainable and economically viable alternative to private car ownership in urban environments [8]. A major obstacle to mass adoption of such vehicle sharing systems is the low reliability of service [4]. In many such systems, the users regularly experience service unavailability: Either there are no available vehicles at the origin station when they decide to rent a vehicle or there is not enough parking spots at the destination station when they decide to return a rented vehicle. In order to provide customer experience that is comparable to the comfort of using a privately-owned vehicle, a sharing system should *guarantee* that a user will always be able to pickup a vehicle at the desired origin location and later return the vehicle at the destination location. Notice that the first type of service failure occurs when a station becomes completely empty, while the second type of service failure occurs when a station becomes completely full. Therefore, a critical task of the system operator is to determine a) the number of vehicles in the fleet and b) a strategy for future redistribution of vehicles between stations that jointly ensure that no station becomes completely empty and no station becomes completely full during the operation of the system.

The authors are with Faculty of Electrical Engineering, Czech Technical University in Prague.

Indeed, in recent years, researchers have increasingly focused on the development of algorithms that support design and operation of vehicle sharing systems [2]. In particular, there is a large body of work focusing on the analysis and synthesis of policies for so-called vehicle rebalancing: Since urban transportation patterns are structurally imbalanced (e.g., in the morning, customers tend to rent vehicles in residential areas and return them in business areas), the vehicles must be continuously moved (rebalanced) from stations with a surplus of vehicles to stations experiencing a shortage of vehicles. In order to determine efficient rebalancing strategies and to quantify the amount of rebalancing needed in a particular vehicle sharing system, researchers proposed fluidic models [5], queue theoretic models [11] as well as heuristics methods [9].

The above methods adopt the "patient customer" assumption, i.e., the customers wait (form a queue) until a vehicle (or a parking space) becomes available. The rebalancing flows are added to the system to ensure that for each station, incoming vehicle flow is on expectation equal to the outgoing vehicle flow. In practice, however, the actual demand flows deviate from expectation which leads to situations when a station temporarily becomes empty or full and the customers are forced to queue. Spieser et al. [10] provide a method for determining a lower bound on the fleet size that is necessary to ensure stability of such queues. In contrast, we assume "impatient customers" and thus our goal is to ensure a high-level of service availability, i.e., we desire that almost all passengers are served immediately. George and Xia [3] studied the relation between the fleet size and the service availability within a queue-theoretic model. Yet, the analysis is limited to systems with time-invariant demand in asymptotic regime when the number of vehicles in the system goes to infinity. In result, to determine an appropriate fleet-size and rebalancing strategy for a real-world vehicle sharing system, a popular approach is to resort to simulation-based optimization [12], [13], [1]. However, the simulation-based approaches are unable to provide guarantees on service availability.

In this paper, we propose a novel rebalancing strategy that is capable of ensuring the desired level of service availability. The customer demand is modeled stochastically as being generated by a Poisson process with a time-varying rate. The proposed rebalancing strategy repeatedly solves a receding-horizon optimization problem to find a rebalancing schedule that ensures that no service failures will occur in future with a given level of confidence. The problem of finding a minimum-cost rebalancing schedule at a given point in time

is formulated as a chance-constrained optimization problem, with the optimization variables representing the number of vehicles to be redistributed between any two stations. We show that this chance-constrained problem can be efficiently solved by conversion into an equivalent linear program. Further, a derivation of a theoretical upper bound on the expected number of service failures is provided. In order to address the problem of fleet sizing, we consider that the system contains a special station called "the depot" that holds an unlimited amount of vehicles that can be used to dynamically increase the number of vehicles circulating in the system. After an initial "warm-up" period, the number of vehicles circulating in the system converges and the converged value can be interpreted as the fleet size that is needed to maintain the required level of service. Finally, our proposed method is evaluated using historical demand from Boston's bike-sharing system. The simulation results demonstrate that the proposed method is capable of achieving the desired level of service availability when facing real-world demand. Moreover, the proposed method can achieve full service availability while making less than half of the rebalancing trips than a state-of-the art rebalancing method we tested on the same scenario.

The paper is structured as follows. In Section II, we formalize the problem. In Section III, we introduce the solution approach, namely the receding horizon algorithm. In Section IV, we discuss the chance-constrained optimization problem and show how it can be efficiently solved. Section V contains a theoretical analysis of our method. In Section VI, we evaluate our algorithm using historical data from Boston's bike sharing system. Section VII concludes the paper.

II. PROBLEM STATEMENT

A vehicle-sharing system consists of a set of stations $S = \{\mathbf{0}, 1, ..., n\}$, where $\mathbf{0}$ denotes the depot. The capacity of station i is denoted ω_i. The depot has infinite capacity.

Users of the system can rent a vehicle from a station i to travel to station j. Formally, a rental request is a tuple (t, i, j), where t is the start time of the trip, i is the start station and j is the destination station. Rental demand is then a set of requests such that the rental requests from station i towards station j are generated by a Poisson process with intensity $\lambda_{ij}(t)$.

We assume the travel time between a pair of stations i and j to be a deterministic quantity denoted as η_{ij}. When a user attempts to rent a vehicle from an empty station, we say the request is dropped. Whenever a user arriving to a station is dropped or a user cannot return a vehicle to a station due to insufficient capacity, we say the station has experienced a service availability failure.

To prevent service availability failures, the fleet manager can move vehicles from stations with a surplus of vehicles to stations with shortage of vehicles. This process is called rebalancing. Analogously to user travel time, the delay associated with rebalancing a vehicle from station i to station j is deterministic and denoted ξ_{ij}.

The fleet operator can increase the number of vehicles in the system by relocating a vehicle from the depot to a chosen station in the system. In order to ensure that new vehicles are only introduced to the system if a shortage of vehicles at a particular station cannot be solved by rebalancing between stations, the cost of moving a vehicle from the depot is assumed to be larger than the cost of rebalancing from the most distant station. This is to incentivize utilizing vehicles that are already present in the system.

Analogously to the customer demand, a rebalancing schedule r is modeled as a sequence of tuples (t, i, j), each representing relocation of a single vehicle in time t from station i to station j. If we associate a cost $c_{ij}(t)$ with moving a single vehicle from i to j in time t, we can define the total rebalancing effort of a schedule r as $\sum_{(t,i,j)\in r} c_{ij}(t)$.

Since the future demand is generated by a Poisson process, any station can experience an arbitrarily high number of travel request with non-zero probability. Therefore, failure-freeness of the system cannot be achieved with certainty. It is, however, reasonable to ask for a system that operates failure free with high probability as stated in the following problem formulation:

Problem 1. Find a control strategy for vehicle rebalancing that ensures that in any time interval of given fixed length, the system will operate failure-free with chosen confidence level z, where $1 - z$ represents the maximum allowed probability of failure in the system during the time interval.

III. SOLUTION APPROACH

Most vehicle sharing systems have two properties that complicate the design of efficient rebalancing policies with guaranteed service availability. Firstly, the system dynamics is highly stochastic, because the vehicles move between stations in response to customer demand that cannot be predicted with certainty. Second, the system suffers from high control-input delay, because it can take tens of minutes for rebalancing vehicles to arrive to their destination after they have been dispatched. Therefore, an efficient control strategy needs to stochastically model the future evolution of the system and respond to all possible service failure scenarios by dispatching rebalancing vehicles ahead of time. To achieve such a behavior, we apply a receding-horizon optimization scheme to control the dispatching of rebalancing vehicles. More specifically, the control algorithm periodically observes the current state of the system and computes a schedule for future rebalancing that ensures failure-freeness with a given confidence level up to the given time horizon. Then, the controller dispatches the vehicles that are scheduled to leave immediately and the process is repeated. The optimization horizon is chosen to be the length of the longest vehicle relocation in the system, which allows the controller to relocate vehicles from the most distant stations (or from the depot) to respond to future service failures predicted by the system model.

In order to simplify the implementation of such a control policy, we discretize the model into time steps of length δ.

Algorithm 1: Receding-horizon rebalancing algorithm. The planning horizon is denoted h, the desired confidence level is denoted z

$k_{\text{now}} \leftarrow 0$; **while** *not terminated* **do**
  $\{v_i\}, \{a_i(t')\} \leftarrow$ observe the state of the system ;
  $r^* \leftarrow$ solve Problem 2 using $k_{\text{now}}, \{v_i\}, \{a_i(t')\}, h, z$;
  for $i, j \in S$ **do**
    dispatch r^*_{ij0} vehicles from station i towards station j;
  end
  sleep until $k_{\text{now}} + \delta$;
  $k_{\text{now}} \leftarrow k_{\text{now}} + 1$;
end

That is, the k-th time step corresponds to the time interval $[k \cdot \delta, (k+1) \cdot \delta)$. Additionally, we adapt the customer travel time η_{ij} and rebalancing travel time ξ_{ij} to be represented in time steps. For example, $\eta_{ij} = 2$ will mean that a system user will need two time steps to reach station j after renting a vehicle at station i.

The pseudocode of the proposed rebalancing method is exposed in Algorithm 1. In each iteration, the current state of the system is observed. More precisely, the method observes 1) the current time step k_{now}, 2) the number of vehicles at each station i denoted v_i and 3) for each station i and each future time step $k' = 0, \ldots, h-1$, the number of vehicles that are on the way and scheduled to arrive to the station before the end of time step k' denoted $a_i(k')$. Then, the algorithm constructs and solves an optimization problem that asks for the minimum-effort rebalancing schedule such that the probability of a service failure occurring in the system is bounded by $1 - z$. The solution is a rebalancing schedule r^* that prescribes how many vehicles should be relocated between any two stations during future time steps $0, \ldots, h-1$. The vehicles that are scheduled to depart immediately, i.e., during the time step 0, are dispatched. The remainder of the rebalancing schedule is discarded. The controller sleeps for one time step and the process is repeated.

IV. OPTIMIZING REBALANCING SCHEDULES

The proposed rebalancing method hinges on the ability to compute a rebalancing schedule that avoids service failures in the system. In this section, we will show that this task can be posed as a chance-constrained optimization problem and efficiently solved by conversion to an equivalent linear program.

The inputs to the minimum-effort rebalancing problem are 1) the current time step k_{now}, 2) the number of vehicles at each station i at the beginning of time step k_{now} denoted v_i, 3) the number of vehicles that have already departed and thus are deterministically known to arrive in the future, with $a_i(k)$ denoting the cumulative number of vehicles that will arrive before the beginning of k-th timestep, i.e., in time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$.

The output of the problem is a rebalancing schedule represented as an $n \times n \times h$ matrix $r = [r_{ijk}]$, where each element r_{ijk} represents the number of vehicles to be dispatched during time step $k - k_{\text{now}}$ from station i to station j.

The vehicle sharing system dynamics is modeled as follows.

Demand: The cumulative number of passenger-carrying vehicles that departed from station i during time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ is a Poisson-distributed random variable denoted as $D_i^{out}(k)$ with mean

$$\mu_i^{out}(k) = \sum_{j \in S} \int_{\delta \cdot k_{\text{now}}}^{\delta \cdot (k_{\text{now}} + k)} \lambda_{ij}(t)\, dt. \tag{1}$$

Similarly, the cumulative number of passenger-carrying vehicles that arrive to station i during time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ is a Poisson-distributed random variable denoted as $D_i^{in}(k)$ with mean $\mu_i^{in}(k) =$

$$\sum_{j \in S} \int_{\delta \cdot k_{\text{now}}}^{\delta \cdot (k_{\text{now}} + k)} \begin{cases} \lambda_{ji}(t - \delta \cdot \eta_{ji}) & \text{if } (t - \delta \cdot \eta_{ji}) \geq \\ & \quad \delta \cdot k_{\text{now}} \\ 0 & \text{otherwise} \end{cases} dt. \tag{2}$$

Note that this definition only considers rental requests that will depart in the future, because the rental requests that have departed in past are already accounted for in $a_i(k)$, i.e., in the state observation variables representing the number of future arrivals to station i.

The change in the number of vehicles during the time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ is a random variable denoted

$$B_i(k) = D_i^{in}(k) - D_i^{out}(k). \tag{3}$$

The difference of two independent Poisson random variables is a Skellam-distributed random variable and thus $B_i(k)$ is a Skellam-distributed random variable with parameters $\mu_1 = \mu_i^{in}(k)$ and $\mu_2 = \mu_i^{out}(k)$.

Rebalancing: The cumulative number of rebalancing vehicles that departed from station i during the time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ is a function of rebalancing schedule $r = \{r_{ijk}\}$ denoted $\rho^{\text{out}}(k; r)$ and defined as

$$\rho_i^{out}(k; r) = \sum_{j \in S} \sum_{k'=0}^{k} r_{ijk}. \tag{4}$$

Similarly, the cumulative number of rebalancing vehicles that have not departed yet but will arrive to station i during time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ is a function of rebalancing schedule $r = \{r_{ijk}\}$ denoted $\rho^{\text{in}}(k; r)$ and defined as

$$\rho_i^{in}(k; r) = \sum_{j \in S} \sum_{k'=0}^{k} \begin{cases} r_{ji(k-\xi_{ji})} & \text{if } (k - \xi_{ji}) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Now we can define the change in the number of vehicles at station i during time interval $[k_{\text{now}} \cdot \delta, (k_{\text{now}} + k) \cdot \delta)$ due to a rebalancing schedule $r = \{r_{ijk}\}$ as

$$\rho_i(k; r) = \rho_i^{in}(k; r) - \rho_i^{out}(k; r). \tag{6}$$

Vehicles: The number of vehicles at station i at the end of time step k under rebalancing schedule r is a random variable denoted by $V_i(k)$ that is defined as

$$V_i(k) = v_i + a_i(k) + B(k) + \rho(k; r), \tag{7}$$

Note that $v_i(k)$, $a_i(k)$, and $\rho(k; r)$ are deterministic quantities, meaning that the randomness of $V_i(k)$ is solely caused by the uncertainty in future demand.

Given a planning horizon h and confidence bound z, we can now formulate the minimum-effort rebalancing problem as a chance-constrained optimization problem:

Problem 2 (Minimum-Effort Rebalancing, Chance-Constrained Program)**.**

$$r^* = \min_{r=[r_{ijk}]} \sum_{k=0}^{h} \sum_{i \in S} \sum_{j \in S} c_{ij}(k) \cdot r_{ijk} \quad \text{subj. to} \tag{8}$$

$$\mathbb{P}[V_i(k; r) \leq 0] \leq 1 - z \qquad \forall k \in \{0, ..., h\} \tag{9}$$

$$\mathbb{P}[V_i(k; r) \geq \omega_i] \leq 1 - z \qquad \forall k \in \{0, ..., h\} \tag{10}$$

$$r_{ijk} \in \mathbb{N}_0 \qquad \forall i, j \in S, k \in \{0, ..., h\}. \tag{11}$$

Objective (13) minimizes the rebalancing effort over the planning horizon. Constraints (9)-(10) ensure that the probability of service failures is within the bounds. In (11), we require each element of the rebalancing schedule to be a nonnegative integer.

Most natural cost function are time-invariant. For example, one could choose the distance or travel time between stations, i.e. $c_i j(\tau) = \xi_{ij}$. The issue with this choice is that under such a cost function, the solution space becomes highly symmetric and thus there may be many optimal rebalancing schedules only differing in how late they move rebalancing vehicles between stations. In practice, however, the fleet manager would prefer to dispatch at the latest possible moment. Dispatching early means that if the policy changes, the vehicles cannot be used until they arrive at their destination. Dispatching late could be achieved, e.g, by adopting the cost function

$$c_{ij}(\tau) = \beta_{ij}(1 - \tau\epsilon), \tag{12}$$

where β_{ij} is a suitable time-invariant cost function (e.g., the distance between the stations) and ϵ is a small positive number. This definition enforces that ties between two otherwise equivalent rebalancing schedules are broken in favor of the one with later departures.

A convenient property of Problem 2 is that this chance-constrained program can be converted into a standard integer linear program by replacing the chance constraints by inequality constraints involving the respective cumulative distribution functions, resulting in the following integer program.

Problem 3 (Minimum-Effort Rebalancing, Integer Program)**.**

$$r^* = \min_{r=[r_{ijk}]} \sum_{k=0}^{h} \sum_{i \in S} \sum_{j \in S} c_{ij}(k) \cdot r_{ijk} \quad \text{subj. to} \tag{13}$$

$$\rho_i(k; r) \geq v_i + a_i(k) + F_i^{-1}(k, z) \quad \forall k \in \{0, ..., h\} \tag{14}$$

$$\rho_i(k; r) \leq \omega_i - (v_i + a_i(k) + F_i^{-1}(k, 1 - z)) \quad \forall k \in \{0, ..., h\} \tag{15}$$

$$r_{ijk} \in \mathbb{N}_0 \qquad \forall i, j \in S, k \in \{0, ..., h\}, \tag{16}$$

where $F_i^{-1}(k, z)$ denotes the inverse cumulative distribution function of $B_i(k)$ at point z. The stochastic constraints (9)-(10) in Problem 2 were replaced by their respective inverse cumulative distribution functions in constraints (14)-(15).

Unfortunately, it is possible that Problem 2 (and consequently also Problem 3) admits no feasible solution. This can occur when the rental demand is higher than what would correspond to the chosen confidence level. For example, consider a system with a station that has ten vehicles in time step 0. Further assume that at confidence level z, the station is expected to need at most five vehicles in time step 0. Now suppose that, although unlikely, all ten vehicles are rented during time step 0. In the following time step 1, the station is again expected to need five vehicles, but has no vehicle left. Assuming that the travel time from the nearest station is greater than one time step, there is no solution to increase the number of vehicles to five in time for the next time step. In other words, there is no schedule that can satisfy constraint (9) and (14), respectively. A similar situation may occur with respect to station capacity constraints.

These situations occur rarely, but in order to ensure that the program remains feasible, we further augment the problem with slack variables that will allow some of the constraints to be violated. In order to discourage the violation of these constraints, the penalty for their violation is set to a sufficiently large positive number M. The augmented integer program is as follows:

Problem 4 (Minimum-Effort Rebalancing, Augmented Integer Program)**.**

$$r^* = \min_{r=[r_{ijk}]} \sum_{k=0}^{h} \sum_{i \in S} \sum_{j \in S} c_{ij}(k) \cdot r_{ijk} + M \sum_{k=0}^{h} \sum_{i \in S} (s_i^L(k) + s_i^U(k)) \quad \text{subj. to} \tag{17}$$

$$\rho_i(k; r) + s_i^L(k) \geq v_i + a_i(k) + F_i^{-1}(k, z) \quad \forall k \in \{0, ..., h\} \tag{18}$$

$$\rho_i(k; r) - s_i^U(k) \leq \omega_i - (v_i + a_i(k) + F_i^{-1}(k, 1 - z)) \quad \forall k \in \{0, ..., h\} \tag{19}$$

$$r_{ijk} \in \mathbb{N}_0 \qquad \forall i, j \in S, k \in \{0, ..., h\}, \tag{20}$$

$$s_i^L(k) \in \mathbb{N}_0, \ s_i^U(k) \in \mathbb{N}_0 \qquad \forall i \in S, k \in \{0, ..., h\}, \tag{21}$$

The objective is augmented with a penalty term that penalizes the use of slack variables $s_i^L(\tau)$ and $s_i^U(\tau)$ and constraints (18)-(19) are augmented with the slack variables,

which effectively turns these constraints into soft constraints. It is easy to see that this program always admits a feasible solution.

V. THEORETICAL ANALYSIS

In this section, we analyze the properties of the proposed rebalancing method. First, we show that the chance-constrained program can be solved efficiently in polynomial time. Second, we derive a bound on the expected number of service availability failures in the system running over longer periods of time.

A. Computational Complexity

In order to be able to deploy Algorithm 1 in practical scenarios, one needs to be able to solve efficiently the integer program stated in Problem4. Fortunately, this can be done by solving the linear relaxation of Problem 4. Since the optimal solution to this linear relaxation is guaranteed to be integral, it is also an optimal solution to the original integer program. We prove this by expressing the linear program in the standard from:

$$\begin{aligned} &\min c^T x \text{ subj. to} \\ &Ax \geq b, \end{aligned} \tag{22}$$

and proving that the constraint matrix A is totally unimodular [**?**]. The constraint matrix of Problem 4 is given by the following matrix:

$$A = \begin{pmatrix} B & I & \mathbf{0} \\ -B & \mathbf{0} & I \\ I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I \end{pmatrix}. \tag{23}$$

To prove that A is totally unimodular, we first show that B is totally unimodular. Indeed, B is a standard incidence matrix, having exactly one 1 and one -1 in each column with other entries being zero. It is well-known that matrices of this form are totally unimodular [7].

Proposition 1. *If M is totally unimodular, then*

$$M' = \begin{pmatrix} M \\ -M \end{pmatrix} \tag{24}$$

is also totally unimodular.

Proof. By the definition of total unimodularity, we have to show that the determinant of every square submatrix P of M' is either 0 or ± 1. This trivially holds if P is fully contained in either M or $-M$. Now, suppose that P contains rows of both M and $-M$. By interchanging the appropriate rows and multiplying rows by -1, we obtain P' that is fully contained in M. Since row permutations and multplying rows by -1 only changes the sign of the determinant, we know that $det(P) = \pm 1$. □

As a consequence of this proposition we know that

$$\begin{pmatrix} B \\ -B \end{pmatrix} \tag{25}$$

is totally unimodular.

Lemma 1. *If a matrix M is totally unimodular then*

$$M' = \begin{pmatrix} M & c \end{pmatrix}, \tag{26}$$

where c is a zero vector containing exactly one 1, is also totally unimodular.

Proof. Similarly to Proposition 1, the lemma holds in the case when P is fully contained in M. Suppose again that P contains a part of column c denoted as c'. By applying the determinant expansion by minors [6] on c', we can see that again $det(P) = \pm 1$. Hence, M' is totally unimodular. □

Lemma 2. *If a matrix M is totally unimodular then*

$$M' = \begin{pmatrix} M \\ c^T \end{pmatrix}, \tag{27}$$

where c^T is a zero vector containing exactly one 1, is also totally unimodular.

Proof. The proof is analagous to the proof of Lemma 1. □

To show that A is totally unimodular we can repeatedly apply Lemma 1 and 2. In result, one can solve Problem 4 in polynomial time by solving its linear relaxation.

B. Number of Failures

In this section, we derive an upper bound on the expected number of service failures when the proposed rebalancing algorithm is applied to control a vehicle sharing system over the period of l time steps. The derived expression can be used to tune the confidence level parameter z used during each receding-horizon optimization step, so that the total number of failures experienced by the system over given time period, say over one day, is kept below an acceptable threshold.

We start by taking a simplifying assumption that Problem 3 is always feasible, i.e, the slack variables s_i^L and s_i^U in Problem 4 are never used. In other words, the system always operates withing the confidence bound z. Then, the probability of a station i operating failure-free in time step k is greater than or equal to z. Let us define $X_i(t)$ as a Bernoulli random variable describing whether a station i is experiencing a service failure during time step k. From the above assumption, we know that $\mathbb{P}[X_i(t) = 0] = z$ and $\mathbb{P}[X_i(t) = 1] = 1 - z$. The sum of independent Bernoulli random variables $\sum X_i(t)$ is a Binomial random variable $Y \sim \text{Binom}(l \cdot n, 1 - z)$, where l is the number of time steps and n is the number of stations. Consequently, the expected number of service failures in the system can be expressed as $E[Y] = l \cdot n \cdot (1 - z)$. As we show in the next section, this bound tends to be tight for values of $z \to 1$.

VI. CASE STUDY: BIKESHARING IN BOSTON

In this section, we demonstrate the applicability of the proposed technique in the context of Boston's bike sharing system formerly known as "The Hubway". This system consisted of roughly 160 stations scattered over the larger Boston area, as shown in Figure 1. The Hubway users often

Fleet size			Rebalancing Effort			Dropped ratio (%)			Capacity violation(%)		
No reb.	EMD	Proposed	No reb.	EMD	Proposed	No reb.	EMD	Proposed	No reb.	EMD	Proposed
574	574	574 (z=0.7)	0	15 552	11 780	37.4	18.1	7.9	6.7	0.0	0.0
711	711	711 (z=0.8)	0	18 413	10 598	32.3	12.9	4.8	9.2	0.0	0.0
874	874	874 (z=0.9)	0	19 579	9 210	27.0	8.8	2.5	13.6	0.0	0.0
1 236	1 236	1 236 (z=0.99)	0	18 990	7 752	18.4	3.9	0.5	19.4	0.0	0.0
1 331	1 331	1 331 (z=0.999)	0	19 881	8 332	17.0	3.2	0.2	20.4	0.0	0.0

TABLE I

COMPARISON OF THE SCENARIO WITHOUT REBALANCING, EMD REBALANCING METHOD, AND THE PROPOSED REBALANCING METHOD. WE TESTED ON FIVE VALUES OF z AS LISTED IN THE TABLE. DROPPED RATIO IS THE PERCENTAGE OF DROPPED TRIPS OUT OF THE TOTAL NUMBER OF DEMANDED TRIPS AND CAN BE VIEWED AS THE PROBABILITY OF FINDING A FREE BIKE TO USE. CAPACITY VIOLATIONS DENOTE THE AVERAGE PERCENTAGE OF BIKES THAT WERE STORED IN STATIONS ABOVE CAPACITY. REBALANCING EFFORT IS THE TOTAL NUMBER OF BIKES MOVED AS A RESULT OF REBALANCING.

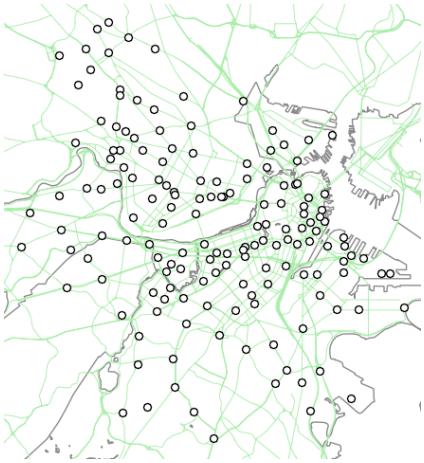

Fig. 1. The stations of Boston's bike sharing system "The Hubway".

faced service availability in the system. As a hypothetical exercise, we simulate the system and use our method to control rebalancing. This allows us to quantify what fleet size and what rebalancing effort would be needed to almost completely avoid service availability failures in the system.

The Hubway released an anonymized historical dataset that contains information about all bike rentals in the system. We use this dataset to estimate the parameters of the system model and to evaluate the performance of different rebalancing strategies in simulation using this historical rental data. More specifically, for our experiment, we extracted records of bike rentals realized during the working days of May 2016. Then, we used this data set to estimate the intensities of demand generating processes $\{\lambda_{ij}\}$ in our model and to estimate the travel times $\{\eta_{ij}\}$ and $\{\xi_{ij}\}$. We simulate the bike sharing system over the period of seven working days using historical requests from the first seven weekdays from June 2016.

In our simulation case study, we compare three rebalancing methods: No rebalancing, EMD rebalancing, and the proposed rebalancing method.

No rebalancing: As a baseline, we simulate the system without rebalancing, i.e., bikes move between station only when a customer rents a bike in one station and returns it in another station.

EMD (Earth mover's distance) rebalancing: [5] is a popular rebalancing method used, e.g., in the Singapore vehicle sharing case study [10]. This approach measures imbalances between the number of incoming and outgoing vehicles at every station per unit of time and periodically dispatches rebalancing vehicles to stabilize the number of vehicles at each station. Similarly to [10], we dispatch rebalancing vehicles once an hour, since more frequent rebalancing leads to significantly increased rebalancing effort with only marginal improvement in service availability.

Proposed rebalancing method: We use Algorithm 1 to control vehicle rebalancing. Further, we add the depot, denoted as $\mathbf{0}$, and set the travel time for rebalancing from depot to $\xi_{i\mathbf{0}} = 1 + \max_{i,j \in S \setminus \mathbf{0}} \{\xi_{ij}\}$. That is, the depot is further away than any other station, which ensures that the fleet size increases only when no other solution exists. The optimization horizon is set to $h = \xi_{\mathbf{0}i} + 1 = 6$ time steps and the length of one time step is chosen as $\delta = 10\,\text{min}$. With increasing horizon, the quality of the solution tends to increase as do the computational requirements.

The experiment is set-up as follows: First, we run the proposed method and record the fleet size and the number of rebalancing vehicles dispatched from the depot during the seven days of the simulation. To allow for easier comparison, we initialize simulation of "no-rebalancing" and EMD methods with the same fleet size that was needed in the proposed method and distribute the fleet uniformly across the stations of the system. During the simulation, we record the state of each vehicle in the system (idle, rented, rebalancing between stations, rebalancing from depot), the number of dropped customer requests, the number of capacity violations, and the number of rebalancing trips.

Table I summarizes the result of the experiment. As we can see, the system that does not use any form of rebalancing suffers from a large number of service availability failures.

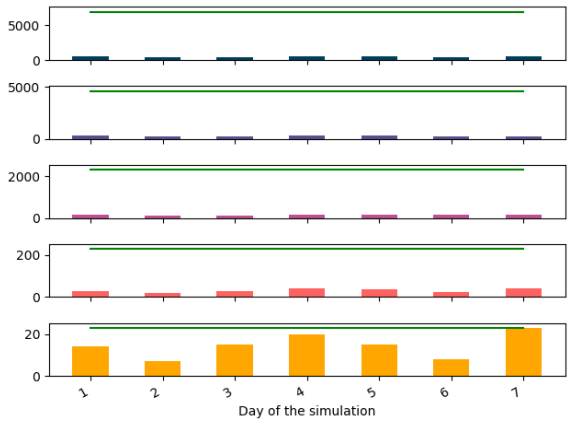

Fig. 2. The number of dropped requests versus the derived upper bound. The five plots show the number of dropped requests during one day, or equivalently the number of service failures since there were no capacity failures, for $z \in \{0.7 \text{ (top)}, 0.8, 0.9, 0.99, 0.999 \text{ (bottom)}\}$. The green line indicates the bound on expected number of failures during one day.

When EMD rebalancing is applied, the number of such failures is greatly reduced at the price of increased rebalancing effort, while the proposed method can further reduce the percentage of dropped requests to nearly zero using less than half the number of rebalancing trips.

Figure 3 illustrates the utilization of the fleet over time when controlled by the proposed method. As the confidence level z increases, the algorithm has to maintain an increasingly larger buffer of vehicles in each station to be able to handle variations in future demand. The size of the fleet is increased only when the extra vehicles are needed to achieve the desired service availability level. We can see that the algorithm initially undergoes a warm-up phase during which many new vehicles are dispatched from the depot and introduced to the system (manifested by the green area preceding the onset of the first morning peak and the first evening peak). After the first day, the fleet size converges to a fixed value. Figure 4 shows the intensity of demand and the number of dropped requests over time for the three tested rebalancing algorithms.

Finally, in Figure 2, we show the total number of service failures during each of the seven days and compare this empirical value with the bound on the number of service failures derived in Section V-B.

VII. CONCLUSION

In this paper, our goal was to design a rebalancing strategy for a vehicle sharing system that ensures full service availability up to the desired confidence level. That is, we desire that almost all users can find a vehicle when they arrive to a station and later on, they can find an available parking spot when returning the vehicle at the destination station.

We model the future evolution of the number of vehicles at each station as a stochastic process. The proposed rebalancing strategy iteratively solves a chance-constrained optimization problem over such a model to ensure that no service failure will occur in the future with given level of confidence. Crucially, we show that such an optimization problem possesses favorable structural properties that allow its solution to be found by conversion to an equivalent linear program.

We demonstrated the practical applicability of the proposed approach in the context of Boston's bike sharing system using historical rental demand. Our simulation shows that the proposed method can indeed prevent service failure with the desired confidence. Further, our simulation experiment revealed that the proposed method is able to achieve almost full service availability while making less than half of the rebalancing trips compared to EMD, a state-of-the-art rebalancing method.

Acknowledgments: This research was supported by the Czech Science Foundation (grant No. 18-23623S) and OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

REFERENCES

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

[2] D. Gavalas, C. Konstantopoulos, and G. Pantziou. Chapter 13 - Design and management of vehicle-sharing systems: A survey of algorithmic approaches. In Mohammad S. Obaidat and Petros Nicopolitidis, editors, *Smart Cities and Homes*, pages 261–289. Morgan Kaufmann, Boston, 2016.

[3] David K. George and Cathy H. Xia. Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research*, 211(1):198–207, May 2011.

[4] Katzev Richard. Car Sharing: A New Approach to Urban Transportation Problems. *Analyses of Social Issues and Public Policy*, 3(1):65–86, December 2003.

[5] Marco Pavone, Stephen L Smith, Emilio Frazzoli, and Daniela Rus. Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, 31(7):839–854, June 2012.

[6] David Poole. *Linear Algebra: A Modern Introduction*. Cengage Learning, Stamford, CT, 4 edition edition, January 2014.

[7] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics. Springer-Verlag, Berlin Heidelberg, 2003.

[8] Shared-use Mobility Center. Shared-use Mobility - Reference Guide. Technical report, Shared-use Mobility Center, October 2016.

[9] Kevin Spieser, Samitha Samaranayake, Wolfgang Gruel, and Emilio Frazzoli. Shared-Vehicle Mobility-on-Demand Systems: A Fleet Operator's Guide to Rebalancing Empty Vehicles. In *Transportation Research Board 95th Annual Meeting*. Transportation Research Board, 2016.

[10] Kevin Spieser, Kyle Treleaven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore. *Road Vehicle Automation (Lecture Notes in Mobility)*, April 2014.

[11] Rick Zhang and Marco Pavone. Control of Robotic Mobility-on-demand Systems. *Int. J. Rob. Res.*, 35(1-3):186–203, January 2016.

[12] Shirley Zhu and Alain Kornhauser. The Interplay Between Fleet Size, Level-of-Service and Empty Vehicle Repositioning Strategies in Large-Scale, Shared-Ride Autonomous Taxi Mobility-on-Demand Scenarios. In *Transportation Research Board 96th Annual Meeting*, 2017.

[13] Tong Zhu. Toward a systematic approach to the fleet size estimation of autonomous mobility-on-demand systems. Msc Thesis, Cornell University, May 2017.

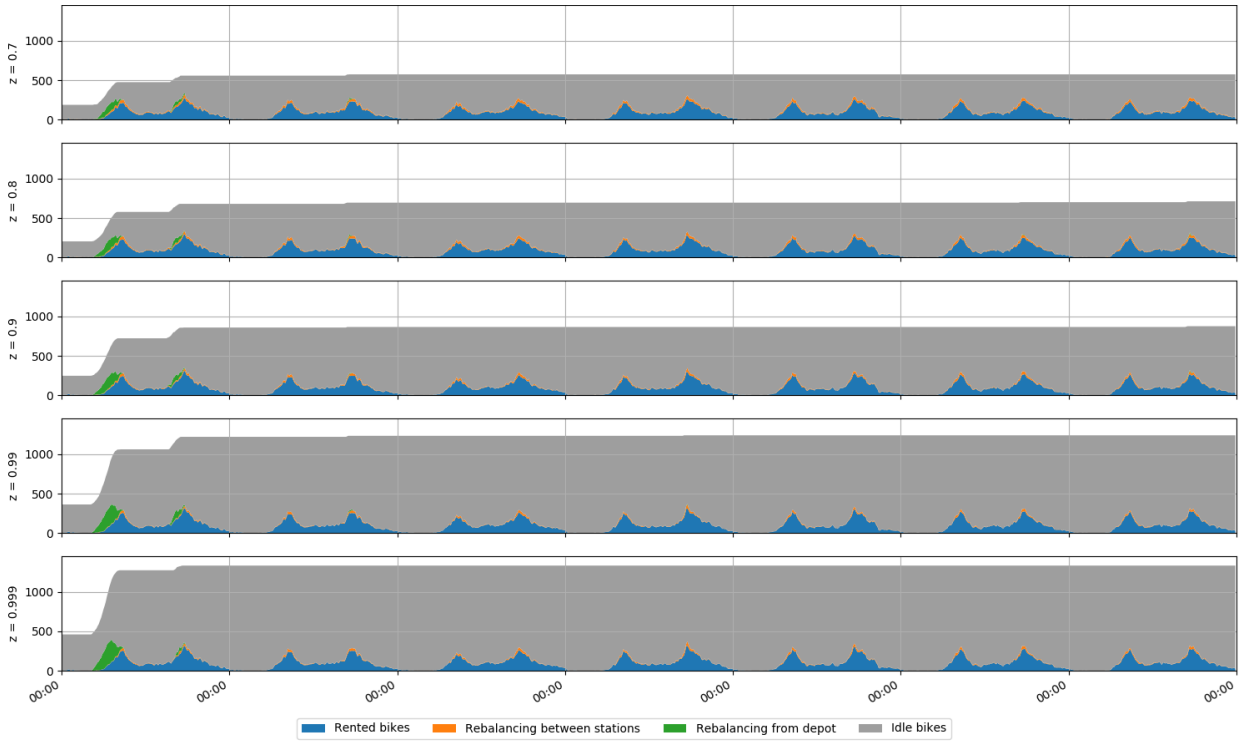

Fig. 3. Area plot showing the status of the fleet vehicles over the seven days of the simulation for different values of $z \in \{0.7 \text{ (top)}, 0.8, 0.9, 0.99, 0.999 \text{ (bottom)}\}$. The blue areas show the number of bikes that are rented by users of the system at each time point. The green area shows the number of bikes that are on the way from the depot towards one of the stations in the system. The orange area shows the number of bikes that are currently being rebalanced between two stations. The gray area denotes the number of vehicles that are parked idle in stations. Therefore, the height of the color area represents the fleet size at a particular point of time.

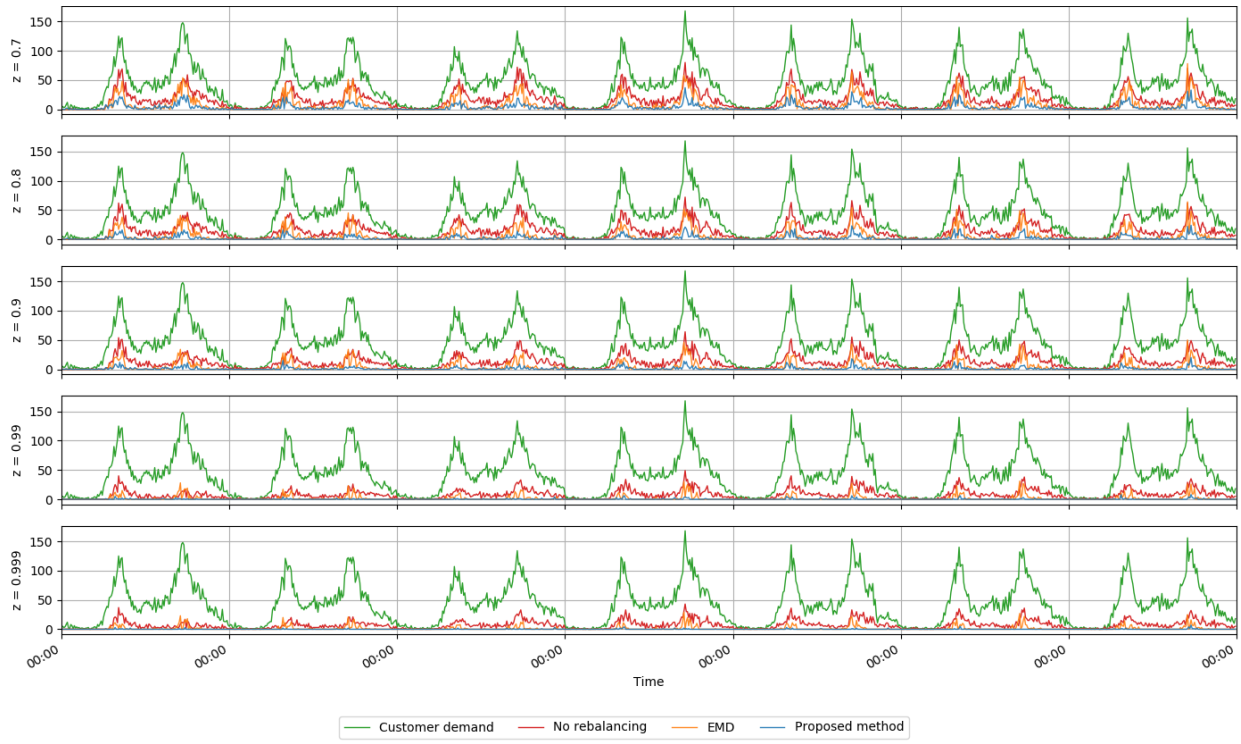

Fig. 4. Comparison of service availability of the three tested algorithms. The green line shows the number of rentals requested during each time step. The red line show the number of requests that were dropped during each time step when no rebalancing was applied. Orange and blue lines, although ovelapping most of the time, show the number of rentals dropped during each timestep for EMD and the proposed method respectively.