

# Minimizing Multimodular Functions and Allocating Capacity in Bike-Sharing Systems<sup>\*</sup>

Daniel Freund, Shane G. Henderson, and David B. Shmoys

Cornell University, Ithaca, NY, USA 14853  
 {df365, sgh9, dbs10}@cornell.edu

**Abstract.** The growing popularity of bike-sharing systems around the world has motivated recent attention to models and algorithms for the effective operation of these systems. Most of this literature focuses on their daily operation for managing asymmetric demand. In this work, we consider the more strategic question of how to (re-)allocate dock-capacity in such systems. We develop mathematical formulations for variations of this problem (service performance over the course of one day, long-run-average performance) and exhibit discrete convex properties in associated optimization problems. This allows us to design a practically fast polynomial-time allocation algorithm to compute optimal solutions for this problem, which can also handle practically motivated constraints, such as a limit on the number of docks moved in the system.

We apply our algorithm to data sets from Boston, New York City, and Chicago to investigate how different dock allocations can yield better service in these systems. Recommendations based on our analysis have been adopted by system operators in Boston and New York City. Beyond optimizing for improved quality of service through better allocations, our results also quantify the reduction in rebalancing achievable through strategically reallocating docks.

## 1 Introduction

As shared vehicle systems, such as bike-sharing and car-sharing, become an integral part of the urban landscape, novel lines of research seek to model and optimize the operations of these systems. In many systems, such as New York City’s Citi Bike, users can rent and return bikes at any location throughout the city. This flexibility makes the system attractive for commuters and tourists alike. From an operational point of view, however, this flexibility leads to imbalances when demand is asymmetric, as is commonly the case. The main contribution of this paper is to identify key questions in the *design* of operationally efficient bike-sharing systems, to develop a polynomial-time algorithm for the associated discrete optimization problems, and to apply this algorithm on real usage data to investigate the effect this optimization can have in practice.

Most bike-sharing systems are dock-based, meaning that they consist of stations, spread across the city, each of which has a number of docks in which bikes can be locked. If a bike is present in a dock, users can rent it and return it at any other station with an open dock. However, system imbalance often causes some stations to have only empty docks and others to have only full docks. In the former case, users need to find alternate modes of transportation, whereas in the latter they might not be able to end their trip at the intended destination. In many bike-sharing systems, this has been found to be a leading cause of customer dissatisfaction (e.g., Capital Bikeshare [2014]).

In order to meet demand in the face of asymmetric traffic, bike-sharing system operators seek to *rebalance* the system by moving bikes from locations with too few open docks to locations with too few bikes. To facilitate these operations, a burst of recent research has investigated models and algorithms to increase their efficiency and increase customer satisfaction. While similar in spirit to some of the literature on rebalancing, in this work we use a different control to increase customer satisfaction. Specifically, we answer the question *how should bike-sharing systems allocate dock capacity to stations within the system so as to minimize the number of dissatisfied customers?* To answer this question, we consider two optimization models, both based on the underlying metric that system performance is captured by the expected number of customers that do not receive service. In the first model, we focus on planning one day, say 6am-midnight, where for each station we determine its allocation of bikes and docks; this framework assumes that there is sufficient rebalancing

<sup>\*</sup> Work supported in part under NSF grants CCF-1526067, CMMI-1537394, CCF- 1522054, and CMMI-1200315.

capacity to restore the desired bike allocation by 6am the next morning. Since in practice this turns out to be quite difficult, the second model considers a set-up induced by a long-run average which assumes that no rebalancing is done overnight. The theory developed in this paper enabled extensive computational experiments on real data-sets; through these we found that there are dock allocations that simultaneously perform well with respect to both models, yielding improvements to both (in comparison to the current allocation) of up to 25%. In Boston and New York City, system operators are adopting improvements based on our models.

**Our Contribution** Raviv and Kolka [2013] defined a *user dissatisfaction function* that measures the expected number of out-of-stock events at an individual bike-share station. To do so, they define a continuous-time Markov chain on the possible number of bikes (between 0 and the capacity of the station). Bikes are rented with rate  $\lambda(t)$  and returned with rate  $\mu(t)$ . Each arrival triggers a change in the state, either decreasing (rental) or increasing (return) the number of available bikes by one. When the number of bikes is 0 and a rental occurs, or equals the station capacity and a return occurs, the customer experiences an out-of-stock event. Using a discrete Markov Chain, they approximate the expected number of out-of-stock events over a finite time-horizon. For fixed rates, the work of Schuijbroek et al. [2017] and O’Mahony [2015] give different techniques to compute the expected number of out-of-stock events exactly. A recursion suggested by Parikh and Ukkusuri [2014] shows that these methods extend to piecewise-constant settings. We use these techniques to compute the expected number of out-of-stock events  $c_i(d_i, b_i)$  that occur over the course of one day at each station  $i$  for a given allocation of  $b_i$  bikes and  $d_i$  empty docks (i.e.,  $d_i + b_i$  docks in total) at station  $i$  at the start of the day.

Given the cost-functions  $c_i(\cdot, \cdot) \forall i$ , our goal is to find an allocation of bikes and docks in the system that minimizes the total expected number of out-of-stock events within a system of  $n$  stations, i.e.,  $\sum_{i=1}^n c_i(d_i, b_i)$ . Since the number of bikes and docks is limited, we need to accommodate a *budget constraint*  $B$  on the number of bikes in the system and another on the number of docks  $D + B$  in the system. Other constraints are often important, such as lower and upper bounds on the allocation for a particular station; furthermore, through our collaboration with Citi Bike in NYC it also became apparent that *operational constraints* limit the number of docks moved from the current system configuration. Thus, we aim to minimize the objective among solutions that require at most some number of docks moved. Via standard dynamic programming approaches, our methods also generalize to other practically motivated constraints, such as lower bounds on the allocation within particular neighborhoods (e.g., in Brooklyn).

We first design a discrete gradient-descent algorithm that provably solves the minimization problem with  $O(n + B + D)$  oracle calls to evaluate cost functions and a (in practice, vastly dominated) overhead of  $O((n + B + D) \log(n))$  elementary list operations. Using scaling techniques and a subtle extension of our gradient-descent analysis, we improve the bound on oracle calls to  $O(n \log(B + D))$ , which still dominate an  $O(\log(B + D)(n \log(n)))$  term for elementary list operations.

The primary motivation of this analysis is to investigate whether the number of out-of-stock events in bikeshare systems can be significantly reduced by a data-driven approach. Through our ongoing collaboration with the operators of New York’s Citi Bike system, it has become evident that current rebalancing efforts overnight are vastly insufficient to realize an optimal (or near-optimal) allocation of bikes for the current allocation of docks, which was designed without any prior knowledge of user demand. In computing an optimal design of the system to address this, the model discussed above still assumes that we can perfectly restore the system to the desired initial bike allocation overnight. Instead, one might consider the opposite regime and focus on optimizing the allocation of docks assuming that no rebalancing occurs at all. To model this, we define an extension of the cost function under a long-run average regime. In this regime, the assumed allocation of bikes at each station is a function of only the number of docks and the estimated demand at that station. Interestingly, our empirical results reveal that bikeshare operators can *have their cake and eat it too*: optimizing dock allocations for one of the objectives (optimally rebalanced or long-run average) yields most of the obtainable improvement for the other. We present the results of these analyses on data-sets from Boston, NYC, and Chicago in Section 7. In the same section, we also provide comparisons of the running-times of the scaling and the naive gradient-descent algorithm (as well as a hybrid of the two) in different regimes.

**Related Work** A recent line of work, including variations by Raviv et al. [2013], Forma et al. [2015], Kaspi et al. [2017], Ho and Szeto [2014], and Freund et al. [2016], considered static rebalancing problems, in which a capacitated truck (or a fleet of trucks) is routed over a limited time horizon. The truck may pick up and drop off bikes at each station, so as to minimize the expected number of out-of-stock events that occur after the completion of the route. These are evaluated by the same objective function of Raviv and Kolka [2013] that we consider as well.

In contrast to this line of work, O’Mahony [2015] addressed the question of allocating both docks and bikes; he uses the user dissatisfaction function (defined over a single interval with constant rates) to design a mixed integer program over the possible allocations of bikes and docks. Our work extends upon this by providing a fast polynomial-time algorithm for that same problem and extensions thereof. The optimal allocation of bikes has also been studied by Jian and Henderson [2015], Datner et al. [2015], and by Jian et al. [2016], with the latter also considering the allocation of docks.<sup>1</sup> They each develop frameworks based on ideas from simulation optimization; while they also treat demand for bikes as being exogeneous, their framework captures the downstream effects of changes in supply upstream. Interestingly, the work of Jian et al. [2016] found that these effects are mostly captured by decensoring piecewise-constant demand estimates.

Orthogonal approaches to the question of where to allocate docks have been taken by Kabra et al. [2015] and Wang et al. [2016]. The former considers demand as endogeneous and aims to identify the station density that maximizes sales, whereas we consider demand and station locations as exogeneously given and aim to allocate docks and bikes to maximize the amount of demand that is being met. The latter aims to use techniques from retail location theory to find locations for stations to be added to an existing system.

Further related work includes a line of work on rebalancing triggered by Chemla et al. [2013]. Subsequent papers, e.g., by Nair et al. [2013], Dell’Amico et al. [2014], Erdoğan et al. [2014], and Erdoğan et al. [2015], solve the routing problem with fixed numbers of bikes that need to be picked up/dropped off at each station – work surveyed by de Chardon et al. [2016]. Other approaches to rebalancing include for example the papers of Liu et al. [2016], Ghosh et al. [2016], Rainer-Harbach et al. [2013], and Shu et al. [2013]. While all of these fall into the wide range of recent work on the operation of bike-sharing systems, they differ from our work in the controls and methodologies they employ.

Finally, a great deal of work has been conducted in the context of predicting demand. In this work, we assume that the predicted demand is given, e.g., using the methods of O’Mahony and Shmoys [2015] or Singhvi et al. [2015]. Further methods to predict demand have been suggested by Li et al. [2015], Chen et al. [2016], and Zhang et al. [2016] among others. Our results can be combined with any approach that predict demand at each station independently of all others.

**Relation to Multimodularity** Our algorithms and analyses exploit the fact that the cost-function  $c(\cdot, \cdot)$  is multimodal (cf. Definition 1) at each station. This provides an interesting connection to the literature on discrete convex analysis. In recent work by Kaspi et al. [2017] it was independently shown that the number of out-of-stock events  $F(b, U - d - b)$  at a bike-share station with fixed capacity  $U$ ,  $b$  bikes, and  $U - d - b$  unusable bikes is  $M$ -natural convex in  $b$  and  $U - d - b$  (see the book by Murota [2003] and the references therein). Unusable bikes effectively reduce the capacity at the station, since they are assumed to remain in the station over the entire time horizon. A station with capacity  $U$ ,  $b$  bikes, and  $U - b - d$  unusable bikes, must then have  $d$  empty docks; hence,  $c(d, b) = F(b, U - d - b)$  for  $d + b \leq U$ , which parallels our result that  $c(\cdot, \cdot)$  is multimodal. Though this would suggest that algorithms to minimize  $M$ -convex functions could solve our problem optimally, one can show that  $M$ -convexity is not preserved, even in the version with only budget constraints.<sup>2</sup> However, since multimodularity is preserved, the techniques of Murota [2004], combined with the submodular function minimization algorithms of Lee et al. [2015], yield an algorithm with running-time guarantee  $O(n^4(D + B))$  to solve the version with only budget constraints.

By exploiting the separability of our objective function (w.r.t. stations) and the associated multimodularity of each station’s cost function, we obtain algorithms with significantly stronger running-times and quickly find solutions for instances at the scale that typically arises in practice.

<sup>1</sup> In fact, the idea behind the algorithm considered by Jian et al. [2016] is based on an early draft of this paper.

<sup>2</sup> In Appendix 9 we provide an example in which a  $M$ -convex function restricted to a  $M$ -convex set is not  $M$ -convex; the example also shows that Murota’s algorithm for  $M$ -convex function minimization can be suboptimal in our setting.

## 2 Model

We denote by  $X = (X_1, \dots, X_s) \in \{\pm 1\}^s$  a sequence of  $s$  customers at a bike-share station. The sign of  $X_t$  identifies whether customer  $t$  arrives to rent or to return a bike, i.e., if  $X_t = 1$  customer  $t$  wants to return a bike and if  $X_t = -1$  customer  $t$  wants to rent a bike. The truncated sequence  $(X_1, \dots, X_t)$  is written as  $X(t)$ . We denote throughout by  $d$  and  $b$  the number of open docks and available bikes at a station before any customer has arrived. Notice that a station with  $d$  open docks and  $b$  available bikes has  $d + b$  docks in total. Whenever a customer arrives to return a bike at a station and there is an open dock, the customer returns the bike, the number of available bikes increases by 1, and the number of open docks decreases by 1. Similarly, a customer arriving to rent a bike when one is available decreases the number of available bikes by 1 and increases the number of open docks by 1. If, however, a customer arrives to rent (return) a bike when no bike (open dock) is available, then she disappears with an out-of-stock event. We assume that only customers affect the inventory-level at a station, i.e., no rebalancing occurs. It is useful then to write

$$\begin{aligned}\delta_{X(t)}(d, b) &:= \max\{0, \min\{d + b, \delta_{X(t-1)} - X_t\}\}, \quad \delta_{X(0)}(d, b) = d \\ \beta_{X(t)}(d, b) &:= \max\{0, \min\{d + b, \beta_{X(t-1)} + X_t\}\}, \quad \beta_{X(0)}(d, b) = b\end{aligned}$$

as a shorthand for the number of open docks and available bikes after the first  $t$  customers.

Our cost function is based on the number of out-of-stock events. In accordance with the above-described model, customer  $t$  experiences an out-of-stock event if and only if  $\delta_{X(t)}(d, b) = \delta_{X(t-1)}(d, b)$ . Since  $d + b = \delta_{X(t)}(d, b) + \beta_{X(t)}(d, b)$  for every  $t$ , this happens if and only if  $\beta_{X(t)}(d, b) = \beta_{X(t-1)}(d, b)$ . Since we are interested in the number of out-of-stock events as a function of the initial number of open docks and available bikes, we can write our cost-function as

$$c^{X(t)}(d, b) = |\{\tau : \tau \leq t, X_\tau = 1, \delta_{X(\tau-1)}(d, b) = 0\}| + |\{\tau : \tau \leq t, X_\tau = -1, \beta_{X(\tau-1)}(d, b) = 0\}|.$$

It is then easy to see that with  $c^{X(0)}(d, b) = 0$ ,  $c^{X(t)}(d, b)$  fulfills the recursion

$$c^{X(t)}(d, b) = c^{X(t-1)}(d, b) + \mathbf{1}_{\{\beta_{X(t)}(d, b) = \beta_{X(t-1)}(d, b)\}}.$$

Given for each station  $i \in [n]$  a distribution, which we call *demand-profile*,  $p_i$  over  $\{(\pm 1)^s, s \in \mathbb{N}_0\}$ , we can write  $c_i(d, b) = \mathbb{E}_{X \sim p_i}[c^X(d, b)]$  for the expected number of out-of-stock events at station  $i$  and  $c(\mathbf{d}, \mathbf{b}) = \sum_i c_i(d_i, b_i)$ . We then want to solve, given budgets  $B$  on the number of bikes and  $D + B$  on the number of docks, a current allocation  $(\bar{\mathbf{d}}, \bar{\mathbf{b}})$ , a constraint  $z$  on the number of docks moved, and lower/upper bounds  $l_i, u_i$  for each station  $i$ , the following minimization problem

$$\begin{aligned}\text{minimize}_{(\mathbf{d}, \mathbf{b})} \quad & \sum_i c_i(d_i, b_i) \\ \text{s.t.} \quad & \sum_i d_i + b_i \leq D + B, \\ & \sum_i b_i \leq B, \\ & \sum_i |(\bar{d}_i + \bar{b}_i) - (d_i + b_i)| \leq 2z, \\ \forall i \in [n] : \quad & l_i \leq d_i + b_i \leq u_i.\end{aligned}$$

Here, the first constraint corresponds to a budget on the number of docks, the second to a budget on the number of bikes, the third to the operational constraints and the fourth to the lower and upper bound on the number of docks at each station. We assume without loss of generality that there exists an optimal solution in which the second constraint holds with equality; to ensure that, we may add a dummy ("depot") station  $\mathcal{D}$  that has  $c_{\mathcal{D}}(\cdot, \cdot) = 0$ ,  $l_{\mathcal{D}} = u_{\mathcal{D}} = B$ , and run the algorithm with a dock-budget of  $D + 2B$ .

In Section 3 we prove that  $c^X(\cdot, \cdot)$  fulfills the following inequalities and is thus multimodular.

**Definition 1 (Hajek [1985], Altman et al. [2000]).** A function  $f : \mathbb{N}_0^2 \rightarrow \mathbb{R}$  with

$$f(d + 1, b + 1) - f(d + 1, b) \geq f(d, b + 1) - f(d, b); \tag{1}$$

$$f(d - 1, b + 1) - f(d - 1, b) \geq f(d, b) - f(d, b - 1); \tag{2}$$

$$f(d + 1, b - 1) - f(d, b - 1) \geq f(d, b) - f(d - 1, b); \tag{3}$$

for all  $d, b$  such that all terms are well-defined, is called multimodular. For future reference, we also define the following implied<sup>3</sup> additional inequalities:

$$f(d+2, b) - f(d+1, b) \geq f(d+1, b) - f(d, b); \quad (4)$$

$$f(d, b+2) - f(d, b+1) \geq f(d, b+1) - f(d, b); \quad (5)$$

$$f(d+1, b+1) - f(d, b+1) \geq f(d+1, b) - f(d, b). \quad (6)$$

Even though we are motivated by the cost-functions defined in this section, our main results hold for arbitrary sums of two-dimensional multimodular functions.

### 3 Multimodularity & An Allocation Algorithm

This section is structured as follows: we first prove that the cost-functions defined in Section 2 are multimodular. Next, we define a natural neighborhood structure on the set of feasible allocations and define a discrete gradient-descent algorithm on this neighborhood structure. We end the section with a proof that solutions that are locally optimal with respect to the neighborhood structure are also globally optimal; this proves that the algorithm finds optimal solutions.

**Lemma 2.**  $c^X(\cdot, \cdot)$  is multimodular for all  $X$ .

*Proof.* We prove the lemma by induction, showing that  $X(t)$  is multimodular for all  $t$ . With  $t = 0$ , by definition,  $c^{X(t)}(\cdot, \cdot) = 0$  and thus there is nothing to show. Suppose that  $c^{X(0)}(\cdot, \cdot)$  through  $c^{X(t-1)}(\cdot, \cdot)$  are all multimodular. We prove that  $c^{X(t)}(\cdot, \cdot)$  is then multimodular as well.

We begin by proving inequality (1). Notice first that if

$$\max\{c^{X(1)}(d+1, b+1), c^{X(1)}(d+1, b), c^{X(1)}(d, b+1), c^{X(1)}(d, b)\} = 0,$$

we can use that inequality (1), by inductive assumption, holds after  $t-1$  customers. Else, we use the inductive assumption on inequality (4) and (5) to prove inequality (1). If  $X_1 = 1$  (and  $d = 0$ ), then both sides of the inequality are 0 and  $\delta_{X(1)}(d+1, b+1) = 0$ ,  $\delta_{X(1)}(d+1, b) = 0$ ,  $\delta_{X(1)}(d, b+1) = 0$ , and  $\delta_{X(1)}(d, b) = 0$ . In that case, we may use the inductive assumption on inequality (5) applied to the remaining  $t-1$  customers. If instead  $X_1 = -1$  (and  $b = 0$ ), then both sides of the inequality are  $-1$  and we have  $\delta_{X(1)}(d+1, b+1) = d+b+2$ ,  $\delta_{X(1)}(d+1, b) = d+b+1$ ,  $\delta_{X(1)}(d, b+1) = d+b+1$ , and  $\delta_{X(1)}(d, b) = d+b$ , so we may apply inequality (4) inductively to the remaining  $t-1$ .

It remains to prove inequalities (2) and (3). We restrict ourselves to inequality (2) as the proof for inequality (3) is symmetric with each  $X_i$  replaced by  $-X_i$  and the coordinates of each term exchanged. As before, if

$$\max\{c^{X(1)}(d-1, b+1), c^{X(1)}(d-1, b), c^{X(1)}(d, b), c^{X(1)}(d, b-1)\} = 0,$$

the inductive assumption applies. If instead  $X_1 = 1$  and the maximum is positive, then the LHS and the RHS are both 0 and we have  $\delta_{X(1)}(d-1, b+1) = 0$ ,  $\delta_{X(1)}(d-1, b) = 0$ ,  $\delta_{X(1)}(d, b) = 0$ ,  $\delta_{X(1)}(d, b-1) = 0$ . In that case, both sides of the inequality are subsequently coupled and the inequality holds with equality.

In contrast, if  $X_1 = -1$  and the maximum is positive, then  $b = 1$ , the RHS is  $-1$ , and the LHS is 0. In this case we have  $\delta_{X(1)}(d-1, b+1) = d$ ,  $\delta_{X(1)}(d-1, b) = d$ ,  $\delta_{X(1)}(d, b) = d+1$ ,  $\delta_{X(1)}(d, b-1) = d$ . Let  $\hat{t}$  denote the next customer such that one of the four terms changes.

If  $X_{\hat{t}} = 1$ , then both terms on the LHS increase by 1, so it remains 0, whereas only the negative term on the RHS increases, so the inequality holds with  $0 \geq -2$ . Moreover, since  $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = 0$ , and  $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = 0$ ; subsequently both sides of the inequality are again coupled.

Finally, if  $X_{\hat{t}} = -1$ , then both terms on the RHS increase by 1 with customer  $\hat{t}$ , but only the negative term on the LHS. Thus, thereafter both sides are again equal. In this case as well, both sides remain coupled thereafter since we have  $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = d+b$ , and  $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = d+b-1$ .  $\square$

**Corollary 3.**  $c_i(\cdot, \cdot)$  is multimodular for any demand-profile  $p_i$ .

*Proof.* The proof is immediate from Lemma 2 and linearity of expectation.  $\square$

<sup>3</sup> (6) and (1) are equivalent, (1) and (2) imply (5), and (3) and (6) imply (4).

### 3.1 An Allocation Algorithm

We next present our algorithm for settings without the operational constraints. Intuitively, in each iteration the algorithm picks one dock and at most one bike within the system and moves them from one station to another. It chooses the dock, and the bike, so as to maximize the reduction in objective value. To formalize this notion, we define the *movement of a dock* via the following transformations.

**Definition 4.** We shall use the notation  $(\mathbf{v}_{-i}, \hat{v}_i) := (v_1 \dots v_{i-1}, \hat{v}_i, v_{i+1} \dots v_n)$ . Similarly,  $(\mathbf{v}_{-i,-j}, \hat{v}_i, \hat{v}_j) := (v_1 \dots \hat{v}_i \dots \hat{v}_j \dots v_n)$ . Then a dock-move from  $i$  to  $j$  corresponds to one of the following transformations of feasible solutions:

1.  $o_{ij}(\mathbf{d}, \mathbf{b}) = ((\mathbf{d}_{-i,-j}, d_i - 1, d_j + 1), \mathbf{b})$  – Moving one open dock from  $i$  to  $j$ ;
2.  $e_{ij}(\mathbf{d}, \mathbf{b}) = (\mathbf{d}, (\mathbf{b}_{-i,-j}, b_i - 1, b_j + 1))$  – Moving a dock & a bike from  $i$  to  $j$ ;
3.  $E_{ijh}(\mathbf{d}, \mathbf{b}) = ((\mathbf{d}_{-i,-h}, d_i - 1, d_h + 1), (\mathbf{b}_{-j,-h}, b_j + 1, b_h - 1))$  – Moving a dock from  $i$  to  $j$  and one bike from  $h$  to  $j$ ;
4.  $O_{ijh}(\mathbf{d}, \mathbf{b}) = ((\mathbf{d}_{-j,-h}, d_j + 1, d_h - 1), (\mathbf{b}_{-i,-h}, b_i - 1, b_h + 1))$  – Moving one bike from  $i$  to  $h$  and one open dock from  $i$  to  $j$  (equivalently, one full dock from  $i$  to  $j$ ).

Further, we define the neighborhood  $N(\mathbf{d}, \mathbf{b})$  of  $(\mathbf{d}, \mathbf{b})$  as the set of allocations that are one dock-move away from  $(\mathbf{d}, \mathbf{b})$ . Formally,

$$N(\mathbf{d}, \mathbf{b}) := \{o_{ij}(\mathbf{d}, \mathbf{b}), e_{ij}(\mathbf{d}, \mathbf{b}), E_{ijh}(\mathbf{d}, \mathbf{b}), O_{ijh}(\mathbf{d}, \mathbf{b}) : i, j, h \in [n]\}.$$

Finally, define the dock-move distance between  $(\mathbf{d}, \mathbf{b})$  and  $(\mathbf{d}', \mathbf{b}')$  as  $\sum_i |(d_i + b_i) - (d'_i + b'_i)|$ .

This gives rise to a very simple algorithm: we first find the optimal allocation of bikes for the current allocation of docks; the convexity of each  $c_i$  in the number of bikes, with fixed number of docks, implies that this can be done greedily by taking out all the bikes and then adding them one by one. Then, while there exists a dock-move that improves the objective, we find the best possible such dock-move and update the allocation accordingly. Once no improving move exists, we return the current solution.

REMARK. A fast implementation of the above algorithm involves six binary heaps for the six possible ways in which the objective at each station can be affected by a dock-move: an added bike, a removed bike, an added empty dock, a removed empty dock, an added full dock, or a removed full dock. In each iteration, we use the heaps to find the best-possible move (in  $O(1)$  time) and update only the values in the heaps that correspond to stations involved. The latter requires a constant number of oracle calls to evaluate the cost functions locally as well as heap-operations that can be implemented in amortized  $O(n \log(n))$  time.

### 3.2 Proof of Optimality

We prove that the algorithm returns an optimal solution by showing that the condition in the while-loop is false only if  $(\mathbf{d}, \mathbf{b})$  minimizes the objective; in other words, if an allocation is locally optimal with respect to  $N(\cdot, \cdot)$  then it is globally optimal. Thus, if the algorithm terminates, the solution is optimal. Before we prove Lemma 7 to establish this, we first define an allocation of bikes and docks as *bike-optimal* if it minimizes the objective among allocations with the same number of docks at each station and prove that bike-optimality is an invariant of the while-loop.

**Definition 5.** Define an allocation  $(\mathbf{d}, \mathbf{b})$  as bike-optimal if

$$(\mathbf{d}, \mathbf{b}) \in \arg \min_{(\hat{\mathbf{d}}, \hat{\mathbf{b}}) : \forall i, d_i + b_i = \hat{d}_i + \hat{b}_i, \sum_i \hat{b}_i = B} \{c(\hat{\mathbf{d}}, \hat{\mathbf{b}})\}.$$

**Lemma 6.** Suppose  $(\mathbf{d}, \mathbf{b})$  is bike-optimal. Given  $i$  and  $j$ , one of the possible dock-moves from  $i$  to  $j$ , i.e.,  $e_{ij}(\mathbf{d}, \mathbf{b}), o_{ij}(\mathbf{d}, \mathbf{b}), E_{ijh}(\mathbf{d}, \mathbf{b})$ , or  $O_{ijh}(\mathbf{d}, \mathbf{b})$ , is bike-optimal. Equivalently, when moving a dock from  $i$  to  $j$ , one has to move at most one bike within the system to maintain bike-optimality.

*Proof.* It is known that multimodular functions fulfill certain convexity properties (see e.g., Murota [2003], Raviv and Kolka [2013]); in particular, for fixed  $d$  and  $b$  it is known that  $c_i(k, d + b - k)$  is a convex function of  $k \in \{0, \dots, d + b\}$ . Thus, if the best allocation out of  $e_{ij}(\mathbf{d}, \mathbf{b})$ ,  $o_{ij}(\mathbf{d}, \mathbf{b})$ ,  $E_{ijh}(\mathbf{d}, \mathbf{b})$ , and  $O_{ijh}(\mathbf{d}, \mathbf{b})$ , was not bike-optimal, there would have to be two stations such that moving a bike from one to the other improves the objective. By the bike-optimality of  $(\mathbf{d}, \mathbf{b})$ , at least one of these two stations must have been involved in the move. We prove that the result holds if  $e_{ij}$  was the best of the set of possible moves  $\{e_{ij}, o_{ij}, E_{ijh}, O_{ijh}\}_{i,j,h \in [n]}$  – the other three cases are almost symmetric. Let  $\ell$  denote a generic third station. Then a bike improving the objective could correspond to one being moved from  $\ell$  to  $j$ , from  $i$  to  $j$ , from  $i$  to  $\ell$ , from  $\ell$  to  $i$ , from  $j$  to  $\ell$  or from  $j$  to  $i$ . In this case, a move from  $\ell$  to  $j$ ,  $i$  to  $j$  and  $i$  to  $\ell$  yield the allocations  $E_{ij\ell}(\mathbf{d}, \mathbf{b})$ ,  $o_{ij}(\mathbf{d}, \mathbf{b})$  and  $O_{ij\ell}(\mathbf{d}, \mathbf{b})$ , respectively. Since  $e_{ij}$  is assumed to be the minimizer among the possible dock-moves, none of these have objective smaller than that of  $e_{ij}(\mathbf{d}, \mathbf{b})$ . It remains to show that moving a bike from  $\ell$  to  $i$ ,  $j$  to  $\ell$  or  $j$  to  $i$  yields no improvement. These all follow from bike-optimality of  $(\mathbf{d}, \mathbf{b})$  and the multimodular inequalities. Specifically, an additional bike at  $i$  yields less improvement and a bike fewer at  $j$  has greater cost in  $e_{ij}(\mathbf{d}, \mathbf{b})$  than in  $(\mathbf{d}, \mathbf{b})$ , since

$$\begin{aligned} c_i(d_i - 1, b_i) - c_i(d_i - 2, b_i + 1) &\leq c_i(d_i, b_i) - c_i(d_i - 1, b_i + 1) \\ c_j(d_j + 2, b_j - 1) - c_j(d_j + 1, b_j) &\geq c_j(d_j + 1, b_j - 1) - c_j(d_j, b_j). \end{aligned}$$

Both of the above inequalities follow from inequality (3).  $\square$

By Lemma 6, to prove optimality of the algorithm, it now suffices to prove that bike-optimal solutions that are locally optimal w.r.t. our neighborhood structure are also global optimal.

**Lemma 7.** *Suppose  $(\mathbf{d}, \mathbf{b})$  is bike-optimal, but does not minimize  $c(\cdot, \cdot)$  subject to budget constraints. Let  $(\mathbf{d}^*, \mathbf{b}^*)$  denote a better (feasible) solution at minimum dock-distance from  $(\mathbf{d}, \mathbf{b})$ . As  $(\mathbf{d}, \mathbf{b})$  is bike-optimal, there exist  $j$  and  $k$  such that  $b_j + d_j < b_j^* + d_j^*$  and  $b_k + d_k > b_k^* + d_k^*$ . Pick any such  $j$  and  $k$ ; then either there exists a dock-move to  $j$  or one from  $k$  that improves the objective.*

*Proof.* The proof of the lemma follows a case-by-case analysis, each of which resembles the same idea:  $(\mathbf{d}^*, \mathbf{b}^*)$  minimizes the dock-move distance to  $(\mathbf{d}, \mathbf{b})$  among solutions with lower function value than  $(\mathbf{d}, \mathbf{b})$ , i.e., among all  $(\mathbf{d}^*, \mathbf{b}^*)$  such that  $\sum_i d_i + b_i = \sum_i d_i^* + b_i^*$ ,  $\sum_i b_i = \sum_i b_i^*$ , and  $c(\mathbf{d}^*, \mathbf{b}^*) < c(\mathbf{d}, \mathbf{b})$ ,  $(\mathbf{d}^*, \mathbf{b}^*)$  has minimum dock-move distance to  $(\mathbf{d}, \mathbf{b})$ . We show that with  $j$  and  $k$  as in the statement of the lemma, either there exists a dock-move to  $j$ /from  $k$  that improves the objective or there exists a solution  $(\mathbf{d}^{**}, \mathbf{b}^{**})$  with objective value lower than  $(\mathbf{d}, \mathbf{b})$ ,  $\sum_i d_i + b_i = \sum_i d_i^{**} + b_i^{**}$ , and  $\sum_i b_i = \sum_i b_i^{**}$ , such that  $(\mathbf{d}^{**}, \mathbf{b}^{**})$  has smaller dock-move distance to  $(\mathbf{d}, \mathbf{b})$ . Since the latter contradicts our choice of  $(\mathbf{d}^*, \mathbf{b}^*)$ , this proves, that in  $(\mathbf{d}, \mathbf{b})$  there must be a dock-move to  $j$ /from  $k$  that yields a lower objective. We distinguish among the following cases:

1.  $d_j < d_j^*$  and  $d_k > d_k^*$ ;
2.  $b_j < b_j^*$  and  $b_k > b_k^*$ ;
3.  $d_j < d_j^*$ ,  $b_k > b_k^*$ , and  $b_j \geq b_j^*$ 
  - (a) and there exists  $\ell$  with  $d_\ell + b_\ell \geq d_\ell^* + b_\ell^*$ ,  $b_\ell < b_\ell^*$ ;
  - (b) and there exists  $\ell$  with  $d_\ell + b_\ell < d_\ell^* + b_\ell^*$ ,  $b_\ell < b_\ell^*$ ;
  - (c) for all  $\ell \notin \{j, k\}$ , we have  $b_\ell \geq b_\ell^*$ , so  $\sum_i b_i > \sum_i b_i^*$ ;
4.  $b_j < b_j^*$ ,  $d_j \geq d_j^*$ ,  $b_k \leq b_k^*$  and  $d_k > d_k^*$ ,
  - (a) and there exists  $\ell$  with  $d_\ell + b_\ell > d_\ell^* + b_\ell^*$  and  $b_\ell > b_\ell^*$ ;
  - (b) and there exists  $\ell$  with  $d_\ell + b_\ell \leq d_\ell^* + b_\ell^*$  and  $b_\ell > b_\ell^*$ ;
  - (c) for all  $\ell \notin \{j, k\}$ , we have  $b_\ell \leq b_\ell^*$ , so  $\sum_i b_i < \sum_i b_i^*$ .

We show that in case (1) a move from  $k$  to  $j$  yields improvement. The proof for case (2) is symmetric. Thus, in cases (3a) and (4a) there exists a move from  $k$  to  $\ell$ , respectively from  $\ell$  to  $j$ , that yields improvement. Since the proofs for cases (3b) and (4b) are also symmetric, we only present the proofs for (3b). Cases (3c) and (4c) contradict our assumption that  $\sum_i b_i = \sum_i b_i^*$  and can thus be excluded. For case (1), we define  $(\mathbf{d}^{**}, \mathbf{b}^{**}) = e_{jk}(\mathbf{d}^*, \mathbf{b}^*)$ , so

$$c((\mathbf{d}^{**}, \mathbf{b}^{**})) - c((\mathbf{d}^*, \mathbf{b}^*)) = (c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*)) + c_k(d_k^* + 1, b_k^*) - c_k(d_k^*, b_k^*).$$

Given that  $\sum_i |d_i - d_i^*| + |b_i - b_i^*| > \sum_i |d_i - d_i^{**}| + |b_i - b_i^{**}|$ , the definition of  $(\mathbf{d}^*, \mathbf{b}^*)$  implies that this difference must be positive. Setting  $(\mathbf{d}', \mathbf{b}') = e_{kj}(\mathbf{d}, \mathbf{b})$ , we bound

$$\begin{aligned} c((\mathbf{d}, \mathbf{b})) - c((\mathbf{d}', \mathbf{b}')) &= \left( c_j(d_j, b_j) - c_j(d_j + 1, b_j) \right) + \left( c_k(d_k, b_k) - c_k(d_k - 1, b_k) \right) \\ &\geq \left( c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*) \right) + \left( c_k(d_k^* + 1, b_k^*) - c_k(d_k^*, b_k^*) \right) = c(\mathbf{d}^{**}, \mathbf{b}^{**}) - c(\mathbf{d}^*, \mathbf{b}^*) > 0. \end{aligned}$$

We prove the inequality between the second and third expression by first showing that

$$c_j(d_j, b_j) - c_j(d_j + 1, b_j) \geq c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*).$$

Applying inequality (3) given in the definition of multimodularity,  $t \geq 0$  times bounds the right-hand side (RHS) by  $c_j(d_j^* - 1 - t, b_j^* + t) - c_j(d_j^* - t, b_j^* + t)$ . Setting  $t = d_j^* - d_j - 1 \geq 0$ , we then find that the RHS is bounded above by

$$c_j(d_j, b_j^* + d_j^* - d_j - 1) - c_j(d_j + 1, b_j^* + d_j^* - d_j - 1).$$

On the other hand, applying inequality (6) repeatedly to the left-hand side (LHS) shows that  $\forall s \geq 0$ , the LHS is at least  $c_j(d_j, b_j + s) - c_j(d_j + 1, b_j + s)$ . Hence, by setting  $s = b_j^* + d_j^* - d_j - b_j - 1$ , which is non-negative since  $b_j + d_j < b_j^* + d_j^*$ , we bound the LHS from below by

$$c_j(d_j, b_j + b_j^* + d_j^* - d_j - b_j - 1) - c_j(d_j + 1, b_j + b_j^* + d_j^* - d_j - b_j - 1).$$

This equals the upper bound on the RHS and thus proves the desired inequality. Similarly, to show

$$c_k(d_k - 1, b_k) - c_k(d_k, b_k) \leq c_k(d_k^*, b_k^*) - c_k(d_k^* + 1, b_k^*), \quad (7)$$

we apply inequality (3)  $d_k - d_k^* - 1$  times to bound the LHS in (7) by  $c_k(d_k^*, b_k + d_k - d_k^* + 1) - c_k(d_k^* + 1, b_k + d_k - d_k^* + 1)$ . Thereafter, we apply inequality (5)  $b_k + d_k - d_k^* + 1 - b_k' \geq 0$  times to obtain the desired bound.

In case (3b), we define  $(\mathbf{d}^{**}, \mathbf{b}^{**}) = E_{jkl}(\mathbf{d}^*, \mathbf{b}^*)$  and  $(\mathbf{d}', \mathbf{b}') = O_{kjl}(\mathbf{d}, \mathbf{b})$ . Similarly to the first case, we need to show that  $c((\mathbf{d}, \mathbf{b})) - c((\mathbf{d}', \mathbf{b}')) \geq c(\mathbf{d}^{**}, \mathbf{b}^{**}) - c(\mathbf{d}^*, \mathbf{b}^*)$ . Since all terms not involving  $j, k$ , and  $\ell$  cancel out and the terms involving  $j$  and  $k$  can be bounded the same way as before, deriving

$$c_\ell(d_\ell, b_\ell) - c_\ell(d_\ell - 1, b_\ell + 1) \geq c_\ell(d_\ell^* + 1, b_\ell^* - 1) - c_\ell(d_\ell^*, b_\ell^*)$$

suffices. We obtain this by repeatedly applying inequalities (3) and (4) to the LHS.  $\square$

## 4 Operational Constraints & Running Time

In this section, we show that the allocation algorithm is optimal for the operational constraints introduced in Section 2 by proving that in  $z$  iterations it finds the best allocation obtainable by moving at most  $z$  docks. We thereby also provide an upper bound on the running-time of the algorithm, since any two feasible dock-allocations can be at most  $D + B$  dock-moves apart. We begin by first formally defining the set of feasible solutions with respect to the operational constraints.

**Definition 8.** Define the  $z$ -dock ball  $S_z(\mathbf{d}, \mathbf{b})$  around  $(\mathbf{d}, \mathbf{b})$  as the set of allocations with dock-move distance at most  $2z$ , i.e.,  $S_0(\mathbf{d}, \mathbf{b}) = \{(\mathbf{d}, \mathbf{b})\}$  and

$$S_z(\mathbf{d}, \mathbf{b}) = S_{z-1}(\mathbf{d}, \mathbf{b}) \cup \left( \bigcup_{(\mathbf{d}', \mathbf{b}') \in S_{z-1}(\mathbf{d}, \mathbf{b})} N(\mathbf{d}', \mathbf{b}') \right).$$

We now want to prove that Lemma 7 continues to hold in the constrained setting; in particular, we show that with the operational constraints as well, local optima are global optima.

**Lemma 9.** With  $(\hat{\mathbf{d}}, \hat{\mathbf{b}}) \in S_z(\mathbf{d}, \mathbf{b}) \setminus S_{z-1}(\mathbf{d}, \mathbf{b})$  bike-optimal and  $c(\mathbf{d}^*, \mathbf{b}^*) < c(\hat{\mathbf{d}}, \hat{\mathbf{b}})$  for some  $(\mathbf{d}^*, \mathbf{b}^*) \in S_z(\mathbf{d}, \mathbf{b}) \setminus S_{z-1}(\mathbf{d}, \mathbf{b})$ , there exists  $(\mathbf{d}', \mathbf{b}') \in S_z(\mathbf{d}, \mathbf{b}) \cap N(\hat{\mathbf{d}}, \hat{\mathbf{b}})$  such that  $c(\mathbf{d}', \mathbf{b}') < c(\hat{\mathbf{d}}, \hat{\mathbf{b}})$ .



*Proof.* Notice that this lemma closely resembles Lemma 7: the sole difference lies in Lemma 7 not enforcing the dock-move to maintain a bound on the distance to some allocation  $(\mathbf{d}, \mathbf{b})$ .

Define  $(\mathbf{d}^*, \mathbf{b}^*)$  as in Lemma 7 with the additional restriction that  $(\mathbf{d}^*, \mathbf{b}^*)$  be in  $S_z(\mathbf{d}, \mathbf{b})$ , i.e., pick a solution in  $S_z(\mathbf{d}, \mathbf{b})$  that minimizes the dock-move distance to  $(\hat{\mathbf{d}}, \hat{\mathbf{b}})$  among solutions with strictly smaller objective value. We argue again that bike-optimality of  $(\hat{\mathbf{d}}, \hat{\mathbf{b}})$  implies that there exist  $j$  and  $k$ , such that  $\hat{d}_j + \hat{b}_j < d_j^* + b_j^*$ , and  $\hat{d}_k + \hat{b}_k > d_k^* + b_k^*$ . Further, for any such  $j$  and  $k$ , we can apply the proof of Lemma 7 to find a move involving at least one of the two that decreases both the objective value and the dock-move distance to  $(\mathbf{d}^*, \mathbf{b}^*)$ .

We aim to find  $j$  and  $k$  such that the move identified, say from  $\ell$  to  $m$ , is guaranteed to remain within  $S_z(\mathbf{d}, \mathbf{b})$ . Notice that  $|\{j\} \cap \{m\}| + |\{k\} \cap \{\ell\}| \geq 1$ . We know that  $d_m^* + b_m^* > \hat{d}_m + \hat{b}_m$  and  $d_\ell^* + b_\ell^* < \hat{d}_\ell + \hat{b}_\ell$ . Suppose the move from  $\ell$  to  $m$  yields a solution outside of  $S_z(\mathbf{d}, \mathbf{b})$ . It follows that  $\hat{d}_m + \hat{b}_m \geq d_m + b_m$  and  $\hat{d}_\ell + \hat{b}_\ell \leq d_\ell + b_\ell$ , so in particular either  $\hat{d}_j + \hat{b}_j \geq d_j + b_j$  or  $\hat{d}_k + \hat{b}_k \leq d_k + b_k$ . Thus, if we can identify  $j$  and  $k$  such that those two inequalities do not hold, we are guaranteed that the identified move remains within  $S_z(\mathbf{d}, \mathbf{b})$ . Define  $k := \arg \max_i \{\hat{d}_i + \hat{b}_i - \max\{d_i + b_i, d_i^* + b_i^*\}\}$ . We can then write

$$\max_i \{\hat{d}_i + \hat{b}_i - \max\{d_i + b_i, d_i^* + b_i^*\}\} \geq \min_i \{1, \max_{i: \hat{d}_i + \hat{b}_i > d_i + b_i} \{(\hat{d}_i + \hat{b}_i) - (d_i^* + b_i^*)\}\}.$$

The latter is at least 1 unless it is the case for all  $i$  that if  $\hat{d}_i + \hat{b}_i > d_i + b_i$  then  $\hat{d}_i + \hat{b}_i \leq d_i^* + b_i^*$ . Thus, unless the above condition fails, we have identified a  $k$  with the required properties. Suppose the condition does fail. Then  $\sum_i d_i + b_i = \sum_i d_i^* + b_i^* = \sum_i \hat{d}_i + \hat{b}_i$  and

$$2z = \sum_i |(d_i + b_i) - (d_i^* + b_i^*)| = \sum_i |(d_i + b_i) - (\hat{d}_i + \hat{b}_i)|$$

imply that for all  $i$  with  $\max\{\hat{d}_i + \hat{b}_i, d_i^* + b_i^*\} > d_i + b_i$ , we have  $\hat{d}_i + \hat{b}_i = d_i^* + b_i^*$ . Thus, it must be the case that  $m$  fulfills  $\hat{d}_m + \hat{b}_m < d_m + b_m$ . The argument for  $j$  is symmetric.  $\square$

By Lemma 9, it suffices to show that the solution found in the  $z$ th iteration is locally optimal among solutions in  $S_z(\mathbf{d}, \mathbf{b})$  to prove the following theorem.

**Theorem 10.** *Starting with a bike-optimal allocation  $(\mathbf{d}, \mathbf{b})$ , in the  $z$ -th iteration, the greedy algorithm finds an optimal allocation among those in  $S_z(\mathbf{d}, \mathbf{b})$ . Hence, the greedy algorithm terminates in at most  $D + B$  iterations.*

*Proof.* We prove the theorem by induction on  $z$ . The base-case  $z = 0$  holds trivially. Suppose in the  $z$ th iteration, the greedy algorithm has found the allocation  $(\mathbf{d}^z, \mathbf{b}^z) \in \arg \min_{(\mathbf{d}^*, \mathbf{b}^*) \in S_z(\mathbf{d}, \mathbf{b})} c(\mathbf{d}^*, \mathbf{b}^*)$ . We need to show that  $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})$  minimizes the cost function among solutions in  $S_{z+1}(\mathbf{d}, \mathbf{b})$ , where

$$(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) := \arg \min_{(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \in N(\mathbf{d}^z, \mathbf{b}^z)} \{c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})\}.$$

We first observe that by Lemma 9, it suffices to show that there is no better solution in  $S_{z+1}(\mathbf{d}, \mathbf{b})$  that is just one dock-move away from  $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})$ . Further, by Lemma 6 and the choice of dock-moves in the greedy algorithm we know that  $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})$  must be bike-optimal. Let  $i$  be the station from which a dock was moved and let  $j$  be the station to which it was moved in the  $z + 1$ st iteration. We denote a third station by  $h$  if the  $z + 1$ st move involved a third one (recall that a dock-move from  $i$  to  $j$  can take an additional bike from  $i$  to a third station  $h$  or take one from  $h$  to  $j$ ). We can then immediately exclude the following cases:

1. Any dock-move in which  $i$  receives a dock from some station  $\ell$ , including possibly  $\ell = j$  or  $\ell = h$ , can be excluded since the greedy algorithm could have chosen to take a dock from  $\ell$  instead of  $i$  and found a bike-optimal allocation (by Lemma 6).
2. The same holds for any dock-move in which a dock is taken from  $j$ .
3. A dock-move not involving either of  $i$ ,  $j$ , and  $h$  yields the same improvement as it would have prior to the  $z + 1$ st iteration. Furthermore, if such a dock-move yields a solution within  $S_{z+1}(\mathbf{d}, \mathbf{b})$ , then prior to the  $z + 1$ st iteration it would have yielded a solution within  $S_z(\mathbf{d}, \mathbf{b})$ . Hence, by the induction assumption, it cannot yield any improvement.

4. A dock-move from station  $i$  (or to  $j$ ), as is implied by the fourth, fifth, and sixth inequality in the definition of multimodularity increases the objective at  $i$  more (decreases the objective at  $j$  less) than it would have prior to the  $z + 1$ st iteration.

We are left with dock-moves either from or to  $h$  as well as dock-moves that involve one of the three stations only via a bike being moved. Suppose that the dock-move in iteration  $z + 1$  was  $E_{ijh}$ ; the case of  $O_{ijh}$  is symmetric. In this case, by inequality (2), a subsequent move of a dock and a bike from  $h$ , i.e.,  $o_{h\ell}$  or  $O_{h\ell m}$  for some  $m$ , increases the objective at  $h$  by at least as much as it did before and can thus be excluded. The same holds for the move of an empty dock to  $h$  (by inequality (3)).

However, subsequent moves of an empty dock from  $h$  (or a full dock to  $h$ ) have a lower cost (greater improvement) and require a more careful argument. Suppose  $e_{h\ell}$  yielded an improvement – the cases for  $E_{h\ell m}$ ,  $o_{h\ell}$ , and  $E_{\ell hm}$  are similar. Notice first that if it were the case that  $d_h^z + b_h^z > d_h + b_h$  and  $d_\ell^z + b_\ell^z < d_\ell + b_\ell$ , then  $e_{h\ell}(E_{ijh}(\mathbf{d}^z, \mathbf{b}^z)) \in S_z(\mathbf{d}, \mathbf{b})$  and has a lower objective than  $(\mathbf{d}^z, \mathbf{b}^z)$  which contradicts the inductive assumption. Furthermore, since it must be the case that  $e_{h\ell}(E_{ijh}(\mathbf{d}^z, \mathbf{b}^z))$  is an element of  $S_{z+1}(\mathbf{d}, \mathbf{b})$  but not of  $S_z(\mathbf{d}, \mathbf{b})$ , it must also follow that either

1.  $d_h^z + b_h^z > d_h + b_h$  and  $d_\ell^z + b_\ell^z \geq d_\ell + b_\ell$  or
2.  $d_h^z + b_h^z \leq d_h + b_h$  and  $d_\ell^z + b_\ell^z < d_\ell + b_\ell$ ,

since otherwise a dock-move from  $h$  to  $\ell$  would either yield a solution in  $S_z$  or one not in  $S_{z+1}$ . Notice further that the inductive assumption implies that  $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \notin S_z(\mathbf{d}, \mathbf{b})$ . Thus, we know that  $d_i^{z+1} + b_i^{z+1} < d_i + b_i$  and  $d_j^{z+1} + b_j^{z+1} < d_j + b_j$ . We can thus argue in the following way about

$$c(e_{h\ell}(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})) - c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) = c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1) + c_\ell(d_\ell^z + 1, b_\ell^z) - c_\ell(d_\ell^z, b_\ell^z).$$

In the first case, since  $o_{h\ell}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$ , the inductive assumption implies that  $c_h(d_h^z, b_h^z - 1) + c_j(d_j^z, b_j^z + 1) \geq c_h(d_h^z, b_h^z) + c_j(d_j^z, b_j^z)$ . Further, by the choice of the greedy algorithm, an additional empty dock at  $\ell$  has no more improvement than an additional dock and an additional bike at  $j$  minus the cost of taking the bike from  $h$ ; otherwise, the greedy algorithm would have moved an empty dock from  $h$  to  $\ell$  in the  $z + 1$ st iteration. Thus,

$$\begin{aligned} c_\ell(d_\ell^z + 1, b_\ell^z) - c_\ell(d_\ell^z, b_\ell^z) &\leq c_j(d_j^z, b_j^z) - c_j(d_j^z, b_j^z + 1) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z) \\ &\leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z, b_h^z) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z) \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1), \end{aligned}$$

implying that  $c(e_{h\ell}(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})) - c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \geq 0$ .

In the second case, since we know that  $e_{i\ell}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$ , the inductive assumption implies  $c_\ell(d_\ell^z + 1, b_\ell^z) + c_i(d_i^z - 1, b_i^z) \geq c_\ell(d_\ell^z, b_\ell^z) + c_i(d_i^z, b_i^z)$ . Further, the choice of the greedy algorithm to take the dock from  $i$ , not  $h$ , implies that  $c_i(d_i^z, b_i^z) - c_i(d_i^z - 1, b_i^z) \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1)$ . Combining these two inequalities again implies that  $e_{h\ell}$  does not yield an improvement.

The remaining cases are ones in which a move only involves  $i, j$ , or  $h$  as the third station that a bike is taken from/added to. Suppose that the transformation in iteration  $z + 1$  was  $E_{ijh}$  – the other cases are similar. A subsequent move of a bike to  $i$  (by inequality (3)) or  $j$  (by inequality (2)) yields at most the improvement that it would have prior to iteration  $z + 1$ . Same holds for taking a bike from  $h$  (by combining inequalities (2) and (3)). Thus, the remaining cases are those in which a bike is taken from  $i$  or  $j$  as well as the ones in which a bike is added to  $h$ .

For a bike taken from  $i$ , notice that the greedy choice was to take a bike from  $h$  rather than from  $i$ , so the increase in objective in taking it from  $i$  now is at least what it was at  $h$  in the  $z + 1$ st iteration. Similarly, since the greedy algorithm chose  $E_{ijh}$  over  $e_{ij}$ , taking the bike from  $j$  has cost at least the cost it had prior to the  $z + 1$ st iteration at  $h$ . But since  $E_{\ell mh}$ , for some  $\ell$  and  $m$  for which it was feasible before the  $z + 1$ st iteration, did not yield an improvement then, it follows that  $E_{\ell mi}$  and  $E_{\ell mj}$  do not yield an improvement after the  $z + 1$ st iteration.

For a bike added to  $h$ , the argument is similar to the one about a dock taken from  $h$  after a bike was taken from  $h$ . For  $O_{\ell mh}$  to be feasible, for some  $\ell, m$  within  $S_{z+1}$ , it must be the case that either  $d_m^z + b_m^z < d_m + b_m$  or  $d_\ell^z + b_\ell^z > d_\ell + b_\ell$ .

In the former case,  $e_{im}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$ , so the inductive assumption implies that the increase in cost of taking a dock from  $i$  in the  $z + 1$ st iteration is at least the decrease realized by moving a dock to  $m$ .

But the increase in objective in taking a dock and a bike from  $\ell$  is at least the increase at  $i$  and  $h$  in the  $z + 1$ st iteration, since otherwise the greedy algorithm would have taken the bike and dock from  $\ell$ . Hence, the decrease in objective at  $h$  and at  $m$  is bounded above by the increase in objective at  $\ell$ .

In the latter case, the inductive assumption implies that an increase in objective at  $\ell$  due to  $O_{\ell mh}$  is bounded below by the increase in objective prior to the  $z + 1$ st iteration due to  $o_{\ell j}$  (since  $o_{\ell j}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$ ). That improvement however is greater-equal to the decrease  $O_{\ell mh}$  yields at  $m$  and at  $h$  combined by the choice of the greedy algorithm in iteration  $z + 1$ . Thus,  $O_{\ell mh}$  cannot yield an improvement.  $\square$

## 5 Scaling Algorithm

We now extend our analysis in Section 4 to adapt our algorithm to a scaling algorithm that provably finds the optimal allocation of bikes and docks in  $O(n \log(B + D))$  iterations. The idea underlying the scaling algorithm is to proceed in  $\lfloor \log_2(B + D) \rfloor + 1$  phases, where in the  $k$ th phase each move involves  $\alpha_k = 2^{\lfloor \log_2(B + D) \rfloor + 1 - k}$  bikes/docks rather than just one. The  $k$ th phase begins by finding the bike-optimal allocation of bikes (given the constraints of only moving  $\alpha_k$  bikes at a time) and terminates when no move of  $\alpha_k$  docks yields improvement. Notice first that the multimodularity of  $c(d, b)$  implies multimodularity of  $c(\alpha_k d, \alpha_k b)$  for all  $k$ . Thus, our analysis in the last two sections implies that in the  $k$ th phase, the scaling algorithm finds the optimal allocation among all that differ in a multiple of  $\alpha_k$  in each coordinate from  $(\bar{\mathbf{d}}, \bar{\mathbf{b}})$ . Further, since  $\alpha_{\lfloor \log_2(B + D) \rfloor + 1} = 1$ , it finds the globally optimal allocation in the final phase. What remains to be shown, is a bound on the number of iterations in each phase. Lemma 12 shows that the number of iterations is  $O(n)$  and thus proves Theorem 11.

**Theorem 11.** *The scaling algorithm finds an optimal allocation in  $O(n \log(B + D))$  iterations.*

**Lemma 12.** *The number of iterations in each phase is no more than  $5n$ .*

*Proof Sketch.* By Theorem 10, the number of dock-moves required in each iteration is the minimum number of dock-moves with which an optimal allocation (for that phase) could be obtained. We try to argue that the dock-move distance between optimal allocations in two subsequent phases cannot be too large. Notice first that if a phase requires  $L > 4n$  dock-moves, then the pigeonhole principle implies that there must exist a sequence of dock-moves of length  $L$  that leads to the same allocation and involves the exact same dock-move twice. For example, if the moves  $e_{ij}$ ,  $e_{kj}$  and  $e_{il}$  occurred, then the moves  $e_{ij}$ ,  $e_{ij}$ , and  $e_{kl}$  yield the same changes, but involve  $e_{ij}$  twice. By Theorem 10, carrying out all of the  $L$  moves except for the two  $e_{ij}$  cannot yield the optimal objective for this phase. Thus, beginning the phase with all but those two moves, we find a suboptimal allocation such that doing the  $e_{ij}$  does yield an optimal allocation; this implies in particular that the  $e_{ij}$  yield improvement at that point. Now, notice that beginning the phase (before moving to a bike-optimal allocation) with the two  $e_{ij}$  moves cannot yield improvement as it gives an allocation that would have been feasible in the last phase.

We now want to bound the improvement of the two moves at the end in terms of the improvement at the beginning. While multimodularity implies diminishing returns in each iteration of the gradient-descent algorithm, this relies on the allocations being bike-optimal. Though the allocation at the beginning of the phase may not be bike-optimal (for the permitted number of bikes to be moved in each iteration of this phase), it cannot be more than  $n$  bike-moves away from being bike-optimal. This allows us to count each dock-move occurring in that phase as either one of the at most  $4n$  moves with no duplicates or as one of the at most  $n$  moves before improvements of subsequent moves are at most what they were prior to moving to bike-optimal. Combining the two bounds, we can derive a contradiction from  $L > 5n$  and thus prove the lemma.  $\square$

## 6 Long-Run-Average Cost

Before we provide the results of our computational study, we define here an extension of the cost-functions considered. One repeated topic that has come up in discussions with operators of bike-share systems is the fact that their means to rebalance overnight do not usually suffice to begin the day with the bike-optimal allocation. In some cities, like Boston, no rebalancing happens at all overnight. Based on these insights, we

define the long-run average of the cost-functions. Given both the allocation of docks  $d_i + b_i$  and the demand profile  $p_i$  of a station  $i$ , we compute the transition probabilities  $\rho_{xy} := \sum_X p_i(X) \mathbf{1}_{\delta_X(d_i + b_i - x, x) = y}$ , that is the probability of station  $i$  having  $y$  bikes at the end of a day, given that it had  $x$  at the beginning. Given those transition probabilities, we define a discrete Markov chain on  $\{0, \dots, d_i + b_i\}$  and define  $\pi_{p_i}^{d_i + b_i}$  as the stationary distribution of that Markov chain.

**Definition 13.** We define the long-run-average cost of station  $i$  with demand profile  $p_i$  as

$$c_i^\pi(d, b) = \sum_{k=0}^{d+b} \pi_{p_i}^{d+b}(k) c_i(d + b - k, k).$$

Notice that (i)  $c^\pi(\cdot, \cdot)$  actually depends only on the sum of its arguments, and (ii) since  $c^\pi(\cdot, \cdot)$  can be seen as the long-run average of evaluating  $c$  over many days, all results proven in the previous sections about  $c$  also extend to  $c^\pi$ .

## 7 Computational Results

In this section we use data from the bike-sharing systems of three major American cities to investigate (i) the effect different allocations of docks might have in each city, and (ii) the regimes under which the scaling and the naive gradient-descent algorithm outperform each other. The three cities, Boston, New York City, and Chicago, vary widely in the sizes of their systems. When the data was collected (summer 2016), Boston had 1300 bikes and 2700 docks across 160 stations, Chicago had 4700 bikes and 9500 docks across 582 stations, and NYC had 6750 bikes and 14840 docks across 447 stations.<sup>4</sup> Despite the differences in size, reallocation could have similarly large effects in the three systems. We summarize our results in Table 1 and explain them throughout this section.

	Present		OPT		150-moved		Moves to OPT	Running Time (Minutes)		
City	$c$	$c^\pi$	$c$	$c^\pi$	$c$	$c^\pi$		Hybrid	Scaling	Greedy
Boston	854	1118	640	943	700	984	407	1.37	1.83	1.44
Chicago	1460	2340	759	1846	1224	2123	1553	5.67	8.78	7.03
NYC	6416	9475	4829	8180	6150	9192	2721	14.02	12.27	18.08

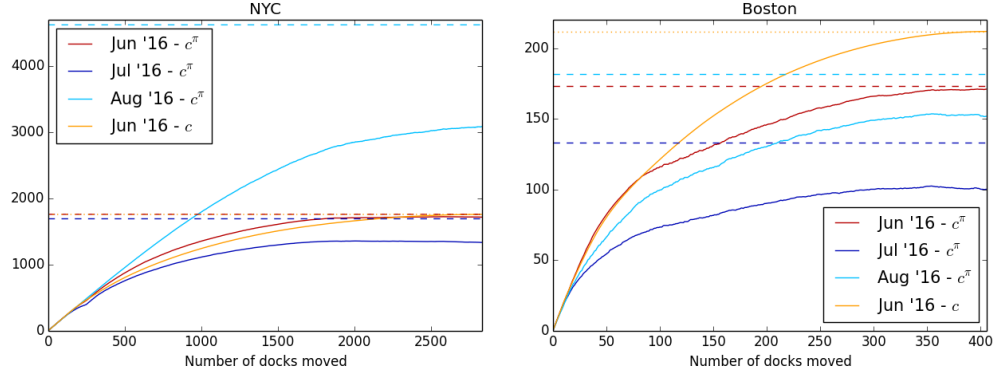
**Table 1.** Summary of main computational results with  $c$  denoting bike-optimal,  $c^\pi$  the long-run-average cost.

**Impact on Objective.** The columns Present, OPT, and 150-moved compare the objective with (i) the present allocation, (ii) the optimal allocation of bikes and docks, and (iii) the best allocation of bikes and docks that can be achieved by moving at most 150 docks from the current allocation. The column headed  $c$  contains the bike-optimal objective for a given allocation of docks, the column headed  $c^\pi$  the long-run-average objective. Various interesting observations can be made. First, while the optimization is done over bike-optimal allocations without regard to the long-run average, it significantly improves that as well in all three cities. Second, in each of the cases, moving 150 docks yields a significant portion of the total improvement feasible to obtain the optimum. This stands in contrast to the large number of moves needed to find the actual optimum (displayed in the column Moves to OPT) and is due to the diminishing returns of the moves.

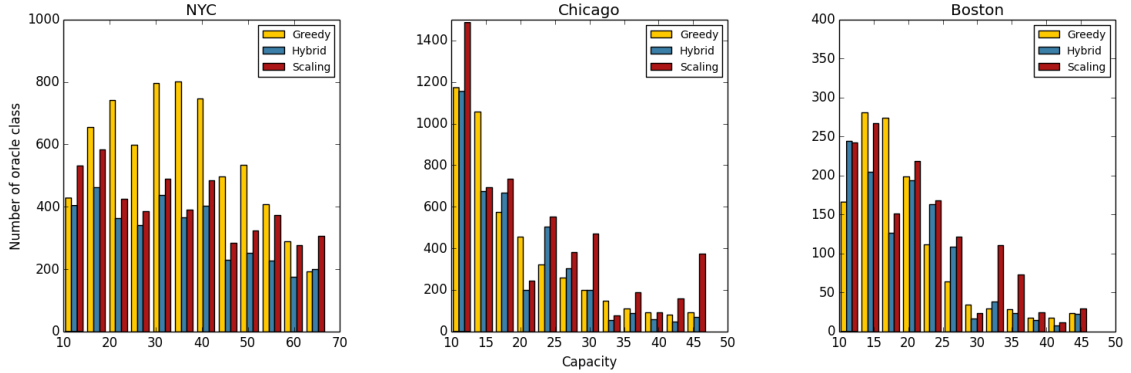
A more complete picture of these insights is given in Figure 1. The x-axis shows the number of docks moved starting from the present allocation, the y-axis shows the improvement in objective. Whereas each x-coordinate in each plot corresponds to a fixed allocation (occurring after that many moves when optimizing the bike-optimal allocation for June 2016 demand estimates), each of the solid lines corresponds to a different way of evaluating it. The dotted line represents the optimum that is achieved through the moves, the dashed lines display the optima for the demand estimates from other months; while these are not achieved through

<sup>4</sup> We remark that these numbers were obtained using the respective JSON feed of each system and do not necessarily capture the entire fleet size, e.g., in New York City a significant number of bikes is kept in depots over night.

the suggested dock moves, significant improvement is made towards them in every case. In particular, the initial moves yield approximately the same improvement for the different objectives/demand estimates. Thereafter, the various improvements diverge, especially for the NYC data from August 2016. This may be partially due to the system expansion in NYC that occurred in the summer of 2016, but does not contradict that all allocations corresponding to values on the x-axis are optimal in the sense of Theorem 10.



**Fig. 1.** Improvement in objective for moves to bike-optimal allocation for June '16 data.



**Fig. 2.** Number of costcurve functions evaluated by each algorithm in each city.

We conclude this part of our analysis with two remarks to contextualize the operational impact our suggestions can have: first, discussions with operators uncovered an additional operational constraint that can arise due to the physical design of the docks. Since these sometimes come in triples or quadruples, the exact moves suggested may not be feasible; e.g., it may be necessary to move docks in multiples of 4. By running the scaling algorithm without the last two iterations, we can find an allocation in which docks are only moved in multiples of 4. With that allocation, the objective of the bike-optimal allocation is 673, 847, and 4896 in Boston, Chicago, and NYC respectively, showing that despite this additional constraint most of the improvements can be realized. Second, it is worth comparing the estimated improvement realized through reallocating docks to the estimated improvement realized through current rebalancing efforts: according to its monthly report, Citi Bike rebalanced an average 3,452 bikes per day in June 2016 (monthly report, NYCBS [2016]), yielding *at most* 3,452 fewer out-of-stock events. Thus, our results show that strategically moving (once), for example, 500 docks diminishes the estimated number of out-of-stock events *by at least as much as a quarter* of Citi Bike's (daily) rebalancing efforts.

**Running Time.** We now compare the measured running times of the greedy and the scaling algorithm. Since computing the individual cost-curves dominates the running-time for each (and that increases with the capacity of the station), we plot in Figure 2 the number of cost-curves for intervals  $d_i + b_i$  that are computed by each algorithm. It is noticeable in Chicago that the scaling algorithm created unnecessary overhead by requiring to computation of new cost-curves for large capacities at many stations that the greedy algorithm did not. This illustrates why the greedy algorithm outperforms the scaling algorithm in both Boston and Chicago (cf. Table 1). In NYC on the other hand, the scaling algorithm performed significantly better than the greedy algorithm. Motivated by this contrast, we implemented a hybrid algorithm that only iterates 8, 4, and 1, rather than all powers of 2. This hybrid outperforms both the greedy and scaling algorithm on all three data-sets. All of the implementations vastly outperform the integer programming based approach suggested by O’Mahony [2015] which evaluates all cost-curves at all stations before solving an integer program; for Boston, Chicago, and NYC, this in itself requires more than 5,900, 22,000, and 30,000 function evaluations (compared to 1,600-6,000 for the hybrid).

## 8 Conclusion

Our cooperation with Citi Bike has given rise to well-motivated problems that combine theory, data, and practice. In this work, we answered an important question in the design of bike-sharing systems, a relatively little studied area compared to rebalancing. The resulting analysis revealed great potential benefits for operators and users, which are already being realized.

## Bibliography

- Eitan Altman, Bruno Gaujal, and Arie Hordijk. Multimodularity, convexity, and optimization properties. *Mathematics of Operations Research*, 25(2):324–347, 2000.
- Capital Bikeshare. Capital Bikeshare member survey report, 2014.
- Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, 2013.
- Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, and Jérémie Jakubowicz Thi-Mai-Trang Nguyen. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 841–852. ACM, 2016.
- Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. Setting inventory levels in a bike sharing network. 2015.
- Cyrille Médard de Chardon, Geoffrey Caruso, and Isabelle Thomas. Bike-share rebalancing strategies, patterns, and purpose. *Journal of Transport Geography*, 55:22–39, 2016.
- Mauro Dell’Amico, Eleni Hadjicostantinou, Manuel Iori, and Stefano Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.
- Güneş Erdoğan, Gilbert Laporte, and Roberto Wolfler Calvo. The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451–457, 2014.
- Güneş Erdoğan, Maria Battarra, and Roberto Wolfler Calvo. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245(3):667–679, 2015.
- Iris A Forma, Tal Raviv, and Michal Tzur. A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological*, 71:230–247, 2015.
- Daniel Freund, Ashkan Norouzi-Fard, Alice Paul, Shane G. Henderson, and David B. Shmoys. Data-driven rebalancing methods for bike-share systems. working paper, 2016.
- Supriyo Ghosh, Michael Trick, and Pradeep Varakantham. Robust repositioning to counter unpredictable demand in bike sharing systems. 2016.
- Bruce Hajek. Extremal splittings of point processes. *Mathematics of operations research*, 10(4):543–556, 1985.
- Sin C Ho and WY Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.
- Nanjing Jian and Shane G Henderson. An introduction to simulation optimization. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1780–1794. IEEE Press, 2015.
- Nanjing Jian, Daniel Freund, Holly M Wiberg, and Shane G Henderson. Simulation optimization for a large-scale bike-sharing system. In *Proceedings of the 2016 Winter Simulation Conference*, pages 602–613. IEEE Press, 2016.
- Ashish Kabra, Elena Belavina, and Karan Girotra. Bike-share systems: Accessibility and availability. *Chicago Booth Research Paper*, (15-04), 2015.
- Mor Kaspi, Tal Raviv, and Michal Tzur. Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. *IIE Transactions*, 49(2):144–158, 2017. doi: 10.1080/0740817X.2016.1224960. URL <http://dx.doi.org/10.1080/0740817X.2016.1224960>.
- Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015.
- Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 33. ACM, 2015.
- Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. 2016.
- Kazuo Murota. *Discrete convex analysis*. SIAM, 2003.
- Kazuo Murota. On steepest descent algorithms for discrete convex functions. *SIAM Journal on Optimization*, 14(3):699–707, 2004.

- Rahul Nair, Elise Miller-Hooks, Robert C Hampshire, and Ana Bušić. Large-scale vehicle sharing systems: analysis of vélib'. *International Journal of Sustainable Transportation*, 7(1):85–106, 2013.
- NYCBS. June 2016 monthly report, 2016.
- Eoin O'Mahony. *Smarter Tools For (Citi) Bike Sharing*. PhD thesis, Cornell University, 2015.
- Eoin O'Mahony and David B Shmoys. Data analysis and optimization for (citi) bike sharing. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 687–694, 2015.
- Pulkit Parikh and Satish Ukkusuri. Estimation of optimal inventory levels at stations of a bicycle sharing system. 2014.
- Marian Rainer-Harbach, Petrina Papazek, Bin Hu, and Günther R Raidl. Balancing bicycle sharing systems: A variable neighborhood search approach. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 121–132. Springer, 2013.
- Tal Raviv and Ofer Kolka. Optimal inventory management of a bike-sharing station. *IIE Transactions*, 45(10):1077–1093, 2013.
- Tal Raviv, Michal Tzur, and Iris A Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, 2013.
- J. Schuijbroek, R.C. Hampshire, and W.-J. van Hoes. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992 – 1004, 2017. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2016.08.029>. URL <http://www.sciencedirect.com/science/article/pii/S0377221716306658>.
- Jia Shu, Mabel C Chou, Qizhang Liu, Chung-Piaw Teo, and I-Lin Wang. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research*, 61(6):1346–1359, 2013.
- Divya Singhvi, Somya Singhvi, Peter I Frazier, Shane G Henderson, Eoin OMahony, David B Shmoys, and Dawn B Woodard. Predicting bike usage for New York citys bike sharing system. Citeseer, 2015.
- Jenhung Wang, Ching-Hui Tsai, and Pei-Chun Lin. Applying spatial-temporal analysis and retail location theory to pubic bikes site selection in Taipei. *Transportation Research Part A: Policy and Practice*, 94: 45–61, 2016.
- Jiawei Zhang, Xiao Pan, Moyin Li, and Philip S Yu. Bicycle-sharing system analysis and trip prediction. *arXiv preprint arXiv:1604.00664*, 2016.

## 9 Connections to $M$ -Convex Functions

In this appendix we first provide the definitions of  $M$ -convex sets and functions, and then show that our cost-function with budget constraints is not  $M$ -convex. For the definitions, it is useful to denote  $\text{supp}^+(\mathbf{x} - \mathbf{y}) = \{i : x_i > y_i\}$ ,  $\text{supp}^-(\mathbf{x} - \mathbf{y}) = \{i : x_i < y_i\}$ , and  $\mathbf{e}_i$  as the canonical unit vector.

**Definition 14 ( $M$ -convex set).** A nonempty set of integer points  $B \subseteq \mathbb{Z}^{2n}$  is defined to be an  $M$ -convex set if it satisfies  $\forall \mathbf{x}, \mathbf{y} \in B, i \in \text{supp}^+(\mathbf{x} - \mathbf{y}), \exists j \in \text{supp}^-(\mathbf{x} - \mathbf{y}) : \mathbf{x} - \mathbf{e}_i + \mathbf{e}_j \in B$ .

**Definition 15 ( $M$ -convex function).** A function  $f$  is  $M$ -convex if for all  $x, y \in \text{dom}(f), i \in \text{supp}^+(\mathbf{x} - \mathbf{y}), \exists j \in \text{supp}^-(\mathbf{x} - \mathbf{y}) : f(x) + f(y) \geq f(x - \mathbf{e}_i + \mathbf{e}_j) + f(y + \mathbf{e}_i - \mathbf{e}_j)$ .

Kaspi et al. [2017] prove a statement equivalent to  $c(\cdot, \cdot)$  being  $M$ -convex. Murota [2004] characterized the minimum of a  $M$  convex function as follows to show that Algorithm 1 minimizes  $M$ -convex functions:

**Lemma 16.** Murota [2003] For an  $M$ -convex function  $f$  and  $x \in \text{dom}(f)$  we have

$$f(x) \leq f(y) \quad \forall y \quad \text{if and only if} \quad f(x) \leq f(x - \mathbf{e}_i + \mathbf{e}_j) \quad \forall i, j.$$

As our example shows, the restriction of  $c$  to the feasible set (with budget constraints) does not guarantee  $M$ -convexity, despite both the set and  $c$  being  $M$ -convex.

*Example 17.* Our example consists of three stations  $i, j$ , and  $k$  with demand-profiles:

$$p_i(-1) = \frac{1}{2}, \quad p_i(+1, -1) = \frac{1}{2}; \quad p_j(+1) = \frac{1}{2}; \quad p_k(+1, -1, -1) = 1.$$



**Algorithm 1**  $M$ -convex function minimization, cf. Murota [2004]

- 
- 1: Find a vector  $x \in \text{dom}(f)$
  - 2: Find  $i, j$  that minimize  $f(x - e_i + e_j)$
  - 3: If  $f(x) > f(x - e_i + e_j)$ , set  $x := x - e_i + e_j$  and go to 2
  - 4: Else, return  $x$
- 

We consider two solutions. In the first,  $i$ ,  $j$ , and  $k$  each have a dock allocated with  $i$  also having a bike allocated, i.e.,  $b'_i = d'_j = d'_k = 1$ , whereas  $d'_i = b'_j = b'_k = 0$  and our budget constraint is  $D = 2, B = 1$ . Then  $c_i(d'_i, b'_i) = \frac{1}{2}$ ,  $c_j(d'_j, b'_j) = 0$ , and  $c_k(d'_k, b'_k) = 1$ . In the second solution,  $d_i^* = b_k^* = d_k^* = 1$ , whereas  $b_i^* = d_j^* = b_j^* = 0$ . Thus, we have  $c_i(d_i^*, b_i^*) = \frac{1}{2}$ ,  $c_j(d_j^*, b_j^*) = \frac{1}{2}$ , and  $c_k(d_k^*, b_k^*) = 0$ , giving that  $1 = c(\mathbf{d}^*, \mathbf{b}^*) < c(\mathbf{d}', \mathbf{b}') = \frac{3}{2}$ . But then the statement of Lemma 16 with  $y = (\mathbf{d}^*, \mathbf{b}^*)$  and  $x = (\mathbf{d}, \mathbf{b})$  implies that, if  $c$  is  $M$ -convex one of  $c((\mathbf{d}'_{-i}, d'_i + 1)$ ,  $c(\mathbf{d}', (\mathbf{b}'_{-i, -k}, b'_i - 1, b'_k + 1))$ , or  $c((\mathbf{d}'_{-i, -j}, d'_i + 1, d'_j - 1), \mathbf{b}')$  must be strictly smaller than  $c(\mathbf{d}', \mathbf{b}')$ . Since this is not the case, we find that  $c$  restricted to the feasible set is not  $M$ -convex, even though the underlying feasible set is  $M$ -convex.