Minimizing Multimodular Functions and Allocating Capacity in Bike-Sharing Systems*

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 their effective operation. 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 (either for service performance over the course of one day or for a long-run-average) and exhibit discrete convex properties in associated optimization problems. This allows us to design a practically fast polynomial-time allocation algorithm to compute an optimal solution 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 led to changes in the system design in Chicago and New York City. Beyond optimizing for improved quality of service through better allocations, our results also provide a metric to compare the impact of strategically reallocating docks and the rebalancing of bikes.

1 Introduction

As bike-sharing systems become an integral part of the urban landscape, novel lines of research seek to model and optimize their operations. In many systems, such as New York City's Citi Bike, users can rent and return bikes at any station within 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 contributions of this paper are 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, to apply this algorithm on real usage data, and to investigate the effect this optimization has in practice.

The largest bike-sharing systems in the US are dock-based, meaning that they consist of stations, spread across a city, each of which has a number of

* Work supported in part under NSF grants CCF-1526067, CMMI-1537394, CCF-1522054, and CCF-1740822, and ARO grant W911NF-17-1-0094.

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*?

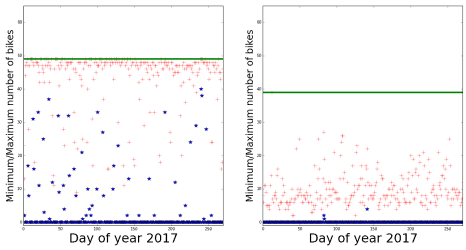

Fig. 1. Minimum (blue) and maximum (red) number of bikes at two bike-share stations over the course of each day of January-September 2017. The green lines denote the capacities of the stations.

The potential in reallocating capacity is easily detected through a superficial analysis of usage data (cf. Figure 1). We give a more theoretically grounded answer to this question by developing 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 capacity overnight 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 happens at all; in a sense, this exhibits the opposite regime. 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 20%. In Chicago and

New York City, system operators have moved hundreds of docks based on these results.

1.1 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 stochastic process on the possible number of bikes (between 0 and the capacity of the station). The stochastic process observes attempted rentals and returns of bikes over time. 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 is attempted, or when it equals the station capacity and a return is attempted, a customer experiences an out-of-stock event. Various follow-up papers, (Schuijbroek et al. [2017], O'Mahony [2015], and Parikh and Ukkusuri [2014]), have suggested different ways 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.

We use the same user dissatisfaction functions to model the question of how to allocate dock capacity within the system. Given $c_i(\cdot, \cdot)\ \forall i$, 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 capacity 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. Notice that D and B could either denote the inventory that is currently present in the system (in which case the question is how to reallocate it) or include new inventory (in which case the question is how to augment the current system design).

After formally defining this model and discussing its underlying assumptions in Section 2, we design in Section 3 a discrete gradient-descent algorithm that provably solves the minimization problem with $O(n + B + D)$ oracle calls to evaluate cost functions and an (in practice, vastly dominated) overhead of $O((n+B+D)\log(n))$ elementary list operations. In Section 4 we show that scaling techniques, together with a subtle extension of the analysis of the gradient-descent algorithm, improve the running-time to $O(n \log(B + D))$ oracle calls and $O(\log(B + D)(n \log(n))$ elementary list operations for the setting without operational constraints; Appendix A.5 explains how operational constraints can be handled when aiming for running-time logarithmic in $B + D$. In Appendix E, we include a computational study to complement this theoretical analysis of the efficiency of our algorithms.

The primary motivation of this analysis is to investigate whether the number of out-of-stock events in bike-sharing systems can be significantly reduced by a data-driven approach. In Section 5, we apply the algorithms to data-sets from Boston, NYC, and Chicago to evaluate the impact on out-of-stock events. One shortcoming of that optimization problem is its assumption that we can perfectly restore the system to the desired initial bike allocation overnight. Through our ongoing collaboration with the operators of systems across the country, it has become evident that current rebalancing efforts overnight are vastly insufficient to realize such an optimal (or even near-optimal) allocation of bikes for the current allocation of docks. Thus, we consider in Section 5.1 the opposite regime, in which 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 operators of bike-sharing systems 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.

Changes implemented by operators, based on our recommendations, allow us to evaluate the impact of our analysis. In Section 6, we prove that observing rentals and returns after capacity has been added provides a natural way to estimate the reduction in out-of-stock events (due to dock capacity added) that can be computed in a very simple manner. Applying that approach to a small set of stations with added capacity in New York City, we derive estimates for the impact changes to the system design have had.

1.2 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. [2016b], 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 rental and return rates) to design a mixed integer program over the possible allocations of bikes and docks. Our work extends upon this by providing a fast algorithm for generalizations of that same problem and extensions thereof. The optimal allocation of bikes has also been studied by Jian and Henderson [2015], Datner et al. [2017], and by Jian et al. [2016], with the latter also considering the allocation of docks (in fact, the idea behind the algorithm considered by Jian et al. [2016] is based on an early draft of this paper). 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.

Jian et al. [2016] found that these effects are mostly captured by decensoring piecewise-constant demand estimates (cf. Section 2.1).

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 literature includes a line of work on rebalancing triggered by Chemla et al. [2013]. Susbequent 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 — de Chardon et al. [2016] extensively surveys these papers. 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 predicts demand at each station independently of all others.

Relation to Discrete Convexity. Our algorithms and analyses crucially exploit the property that the user dissatisfaction functions $c_i(\cdot,\cdot)$ at each station are multimodular (cf. Definition 1). This provides an interesting connection to the literature on discrete convex analysis. In concurrent work by Kaspi et al. [2017] it was 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 $M^\natural$ (read 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 multimodular. Though this would suggest that algorithms to minimize $M^\natural$-convex functions could solve our problem optimally, one can show that $M^\natural$-convexity is not preserved, even in the version with only budget constraints: we provide in Appendix B an example that shows both that an $M^\natural$-convex function restricted to an $M^\natural$-convex set is not $M^\natural$-convex and that Murota's algorithm for $M^\natural$-convex function minimization can be suboptimal in our setting. In fact, when including the operational constraints even discrete midpoint convexity, a strict generalization of multimodularity studied for example by Fujishige and Murota [2000] and Moriguchi et al. [2017], which is in turn

much weaker than $M^\natural$ convexity, breaks down. We provide an example for this in Appendix C. Surprisingly, we are nevertheless able to design fast algorithms; these exploit not only the multimodularity of each individual c_i, but also the separability of the objective function (w.r.t. the stations), that is, the fact that each c_i is only a function of d_i and b_i. This not only extends ideas from the realm of unconstrained discrete convex minimization to the constrained setting, but also yields algorithms that (for our special case) have significantly faster running times than those that would usually arise in the context of multimodular function minimization. Since the conference version of this paper appeared, Shioura [2018] has taken our work as motivation to study M-convex function minimization under L1-distance constraints, a strict generalization of our objective. Finally, Shioura (private communication) pointed out an error in a preliminary version of this paper, and so, although all of the main elements of our proof of correctness of the discrete gradient-descent algorithm can be found in our preliminary version Freund et al. [2017, 2016a], the presentation here differs from that given earlier.

2 Model

In this section we formally define our model and discuss and justify the assumptions underlying it. We begin by giving a formal definition of Raviv and Kolka's user dissatisfaction function.

User Dissatisfaction Function We denote by $X = (X_1, \ldots, X_s) \in \{\pm 1\}^s$ a sequence of s customers at a bike-share station. The sign of X_t identifies whether customer t arrives to rent ($X_t = -1$) or to return a bike ($X_T = 1$). The truncated sequence $(X_1, \ldots, X_t)$ 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. 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 instead 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

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

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

Our objective 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 $\delta_{X(t)}(d,b) = \delta_{X(t-1)}(d,b)$, that is, out-of-stock events occur if and only if an arriving customer does not change the number of bikes at the station. 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 $c^X(d,b) = |\{\tau : X_\tau = 1, \delta_{X(\tau-1)}(d,b) = 0\}| + |\{\tau : X_\tau = -1, \beta_{X(\tau-1)}(d,b) = 0\}|$. Starting with $c^{X(0)}(d,b) = 0$, $c^{X(t)}(d,b)$ then fulfills the recursion $c^{X(t)}(d,b) = c^{X(t-1)}(d,b) + \mathbf{1}_{\{\beta_{X(t)}(d,b)=\beta_{X(t-1)}(d,b)\}}$.

Optimization Problem Given for each station $i \in [n] := \{1, 2, \ldots, n\}$ a distribution p_i over possible sequences of arrivals $\{(\pm 1)^s, s \in \mathbb{N}_0\}$, which we call the *demand-profile*, 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 $c(\boldsymbol{d},\boldsymbol{b}) = \sum_i c_i(d_i,b_i)$. We then want to solve, given budgets B on the number of bikes and $D+B$ on the number of docks, a current allocation $(\bar{\boldsymbol{d}},\bar{\boldsymbol{b}})$, 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

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

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 first two constraints hold 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 $D + 2B$. Here, D may include docks that are currently part of the system as well as inventory that is meant to be added to the system. In fact, in Appendix D we show that we can also optimally solve an optimization problem that involves an additional trade-off between the size of D and the size of z, i.e., that trades off the costs of additional docks with moving docks.

2.1 Discussion of Assumptions

The formulation of our model is based on several key assumptions. Before describing and analyzing the algorithm we use to solve the optimization problem in Section 3, we discuss here the assumptions as well as the advantages that come along with them.

Seasonality and Frequency of Reallocations. In contrast to bike rebalancing, the reallocation of docks is a strategic question that involves docks being moved at most on an annual level. As such, one concern would be that the recommendations for a particular month do not yield improvement for other times

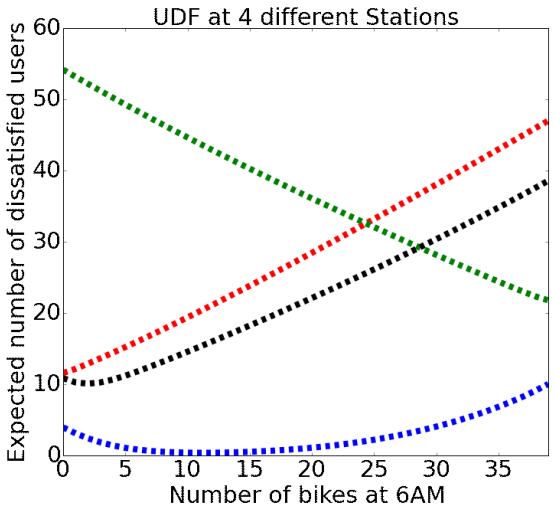

Fig. 2. Visualizations of user dissatisfaction functions, based on real data, as a function of bikes for stations with capacity 39.

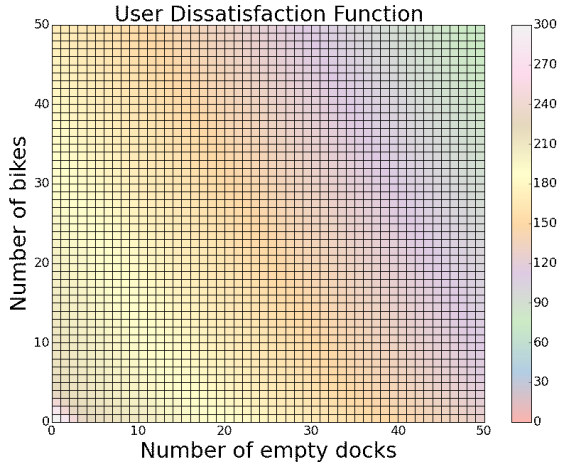

Fig. 3. Visualization as a function of (d, b) at a single station.

of the year. Of course, one way of dealing with this would be to explicitly distinguish, in the demand profiles, between different seasons, i.e., have k different distributions for k different types of days and then consider the expectation over these as the objective. Though the user dissatisfaction functions accommodate that approach, we find on real data (cf. Section 5.3) that this is not actually necessary: the reallocations that yield greatest impact for the summer months of one year also perform very well for the winter months of another. This even held true in New York City, where the system significantly expanded year-over-year: despite the number of stations in the system more than doubling and total ridership increasing by around 70% from 2015 to 2017, we find that the estimated improvement due to reallocated docks is surprisingly stable across these different months.

Cost of Reallocations. Throughout the paper, we consider reallocations as being bounded by the number of docks that are moved instead of associating it with an explicit cost. Mathematically, this is equivalent to handling a fixed per-unit cost for each dock reallocated. This is mostly motivated by the real-life operations of our industry partner: the cost of physically reallocating capacity from one location to another is negligible when compared to the administrative effort (a negotiation with city officials and other stakeholders) needed to reallocate capacity. In particular, this implies that the tactical question of how to carry out the reallocations is of minor importance in practice. Further, the cost of reallocating docks can be compared to the cost of rebalancing bikes: while the (one-off) reallocation of a single dock is about an order of magnitude more expensive than that of a single bike, the reallocated dock has daily impact on improved service levels (in contrast to the one-off impact of a rebalanced bike). Thus, the cost amortizes extremely fast; Citi Bike estimates in as little as 2 weeks. Finally, the cost to acquire new docks is orders of magnitudes higher than all of the aforementioned costs, leading us to focus only on reallocated capacity in our analysis; nevertheless, we show in Appendix D that the algorithm also extends to capture the tradeoff between installing newly bought and reallocating existing docks.

Bike Rebalancing. The user dissatisfaction functions assume that no rebalancing takes place over the course of the planning horizon. System data indicates that this is close to reality at most stations; for example, in New York City (cf. Figure 4), more than 60% of all rebalancing is concentrated at 28 of 762 stations which justifies the assumption for the vast majority of stations at which very little rebalancing happens. For the remaining few stations, at which almost all rebalancing is concentrated, we argue that the no-rebalancing assumption only underestimates the effect of added capacity. Unsurprisingly, none of these stations are identified by the optimization as having their capacity reduced.

Though we assume that no rebalancing occurs over the course of the planning horizon, the optimization model assumes that the initial number of bikes at each station is optimally allocated. We relax this assumption in Section 5.1 when we consider a regime in which no rebalancing occurs at all. Despite the fact that the two regimes can be viewed as polar opposites (optimally rebalanced

overnight and no rebalancing overnight), our results indicate that they yield very similar recommendations for the operators. Our motivation to focus on these opposite extremes is simple: modeling a modest amount of rebalancing poses significant challenges. For example, unlike the effect of daily usage patterns, overnight rebalancing is affected by greater variability from external factors, ranging from the number of trucks to the supply of just-repaired bikes.

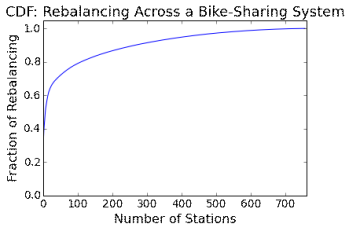

Fig. 4. Fraction of rebalancing actions (bikes being added or taken) at stations within Citi Bike's system. Most rebalancing happens at a small fraction of stations.

Out-of-stock Events and Demand Profiles. In practice, we cannot observe attempted rentals at empty stations nor can we observe attempted returns at full stations. Worse still, given that most bike-sharing systems have mobile apps that allow customers to see real-time information about the current number of bikes and empty docks at each station, there may be customers who want to rent a bike at a station, see on the app that the station has only one bike available presently, and decide against going to the station out of concern that by the time they get there, the bike has already been taken by someone else — should such a case be considered an out-of-stock event (respectively, an attempted rental)? The user dissatisfaction functions assume that such events do not occur as the definition relies on out-of-stock events occurring only when stations are either entirely empty or entirely full.

Further, in order to compute the user dissatisfaction functions, we need to be able to estimate the demand profiles: using only observed rentals and returns is insufficient as it ignores latent demand at empty/full stations. To get around this, we mostly apply a combination of approaches by O'Mahony and Shmoys [2015], O'Mahony et al. [2016], and Parikh and Ukkusuri [2014]: we estimate Poisson arrival rates (independently for rentals and returns) for each 30 minute interval and use a formula developed by O'Mahony et al. [2016] to compute, for any initial condition (in number of bikes and empty docks) the expected number of out-of-stock events over the course of the interval. We plug these into a stochastic recursion suggested by Parikh and Ukkusuri [2014] to obtain the expected number of out-of-stock events over the course of a day as a function of the number of bikes and empty docks at 6AM. This is far from being the only approach to compute user dissatisfaction functions; for example, in Section 6 we

explicitly combine empirically observed arrivals with estimated rates for times when rentals/returns are censored at empty/full stations.

Exogeneous Rentals and Returns. The demand profiles assume that the sequences of arrivals are exogeneous, i.e., there is a fixed distribution that defines the sequence of rentals and returns. Before justifying this assumption, it is worth considering a setting in which it fails spectacularly: consider an allocation of bikes and docks that allocates no bikes at all. Of course, this would imply that no attempted rental is ever successful and therefore no returns ever occur. As such, the sequence of arrivals of returns at one station are not independent of the allocations elsewhere.

The main justification for this assumption comes from work by Jian et al. [2016]; this paper used a simulation optimization approach to find the configuration of bikes and docks across the system that minimizes the number of out-of-stock events over the course of the day. In contrast to the user dissatisfaction functions, decensoring the demand data for the simulation required additional modeling decisions. However, the simulation allowed for two different kinds of endogeneity:

- A return at one station had to be triggered by rentals at another. In particular, each (successful) rental caused a later attempt for a bike to be returned.
- A failed return at a station (due to out-of-stock events, i.e., stations being full) triggered a later attempt to return a bike at a station nearby.

While this simulation approach still assumed that demand for rentals is exogeneous, it endogenized returns, excluding (at least) the example suggested above. However, it causes the resulting simulation optimization problem to be non-convex in an unbounded fashion, that is, one can construct examples, admittedly highly contrived, in which two initial conditions are only *two bikes away from each other* (meaning they differ only by one station having two bikes fewer, the other two more); yet, one of them has arbitrarily many out-of-stock events fewer (in expectation), than the other. Even worse, both solutions still have strictly better objectives than the solution in between (in which one bike is moved). This level of sensitivity not only makes it harder to optimize, but also makes solutions difficult to interpret. Jian et al. [2016] proposed a range of different gradient-descent algorithms as heuristics to find good solutions, including an adaptation of the algorithm we present and analyze in Section 3. Despite the simulation adding key complexities to the system, the heuristics gave only limited improvements – approximately 3% – when given the solution found by the algorithm in Section 3 as a starting point.

Advantages of User Dissatisfaction Functions. The user dissatisfaction functions yield several advantages over a more complicated model such as the simulation. First, they provide a computable metric that can be used for several different operations: in Section 3 we show how to optimize over them for reallocated capacity and in Section 6 we use them to evaluate the improvement from already reallocated capacity. Chung et al. [2018] used them to study an incentive program operated by Citi Bike in New York City, and they have been used extensively for motorized rebalancing (cf. Section 1.2). As such, the user

dissatisfaction functions provide a single metric on which to evaluate different operational efforts to improve service quality, which adds value in itself. Second, for the particular example of reallocating dock capacity that we study here, they yield a tractable optimization problem, which we prove in Section 3. Third, for the reallocation of dock capacity, the discrete convexity properties we prove imply that a partial implementation of the changes suggested by the optimization (cf. Section 5) is still guaranteed to yield improvement. Finally, given a solution to the optimization problem, it is easy to track the partial contribution to the objective from changed capacity at each station, making solutions interpretable.

3 A Discrete Gradient-Descent Algorithm

We begin this section by examining the mathematical structure of the user dissatisfaction functions that allows us to develop efficient algorithms for our optimization problem. To do so, we first define multimodularity.

Definition 1 (Hajek 1985, Altman et al. 2000). *A function $f : \mathbb{N}_0^2 \to \mathbb{R}$ 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 additional inequalities (6 and 1 are equivalent, 1 and 2 imply 5, and 3 and 6 imply 4):*

$$f(d+2,b) - f(d+1,b) \geq f(d+1,b) - f(d,b); \tag{4}$$
$$f(d,b+2) - f(d,b+1) \geq f(d,b+1) - f(d,b); \tag{5}$$
$$f(d+1,b+1) - f(d,b+1) \geq f(d+1,b) - f(d,b). \tag{6}$$

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

By linearity of expectation, Lemma 2 immediately implies that the user dissatisfaction function $c_i(\cdot,\cdot)$ is multimodular for any demand-profile p_i. The proof of the lemma is based on a coupling argument, and appears in Appendix A.1.

In Section 3.1, we define a natural neighborhood structure on the set of feasible allocations and define a discrete gradient-descent algorithm on this neighborhood structure. We prove in Section 3.2 that for the problem without operational constraints solutions that are locally optimal with respect to the neighborhood structure are also globally optimal; since the algorithm only terminates when finding a local optimum, this proves that it returns a globally optimal solution. Finally, in Section 3.3, we prove that the algorithm takes z iterations to find the best allocation obtainable by moving at most z docks; this not only proves that the gradient-descent algorithm optimally solves the minimization problem when including operational constraints, but also guarantees that doing so requires at most $O(D+B)$ iterations.

3.1 Algorithm

We now present our algorithm before analyzing it 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 3. *We shall use the notation* $(\boldsymbol{v}_{-i}, \hat{v}_i) := (v_1 \dots v_{i-1}, \hat{v}_i, v_{i+1} \dots v_n)$. *Similarly,* $(\boldsymbol{v}_{-i,-j}, \hat{v}_i, \hat{v}_j) := (v_1 \dots \hat{v}_i \dots \hat{v}_j \dots v_n)$. *Then a* dock-move *from* i *to* j *corresponds to one of the following transformations of feasible solutions:*

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

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

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

Finally, define the dock-move distance *between* $(\boldsymbol{d}, \boldsymbol{b})$ *and* $(\boldsymbol{d}', \boldsymbol{b}')$ *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 the involved stations. 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 Optimality without Operational Constraints

We prove that the algorithm returns an optimal solution by showing that an allocation $(\boldsymbol{d}, \boldsymbol{b})$ that is locally optimal with respect to $N(\cdot, \cdot)$ must also be

globally optimal. Thus, when the algorithm terminates, the solution returned is optimal. Before we prove Lemma 6 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 state a Lemma that bike-optimality is an invariant of the algorithm; the proof if provided in Appendix A.2.

Definition 4. *An allocation* $(\boldsymbol{d}, \boldsymbol{b})$ *is* bike-optimal *if*

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

Lemma 5. *Suppose* $(\boldsymbol{d}, \boldsymbol{b})$ *is bike-optimal. Given* i *and* j*, one of the possible dock-moves from* i *to* j*, i.e.,* $e_{ij}(\boldsymbol{d}, \boldsymbol{b}), o_{ij}(\boldsymbol{d}, \boldsymbol{b}), E_{ijh}(\boldsymbol{d}, \boldsymbol{b})$*, or* $O_{ijh}(\boldsymbol{d}, \boldsymbol{b})$*, 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.*

By Lemma 5, to prove optimality of the algorithm, it suffices to prove that bike-optimal solutions that are locally optimal w.r.t. our neighborhood structure are also globally optimal. This is formalized in Lemma 6, the proof of which we defer to Appendix A.3. That appendix also contains the proof of Lemma 7, which is a corollary of Lemma 6.

Lemma 6. *Suppose* $(\boldsymbol{d}, \boldsymbol{b})$ *is bike-optimal, but does not minimize* $c(\cdot, \cdot)$ *subject to budget constraints. Let* $(\boldsymbol{d^\star}, \boldsymbol{b^\star})$ *denote a better solution. Then there exists a dock-move that simultaneously reduces the dock-move distance from* $(\boldsymbol{d}, \boldsymbol{b})$ *to* $(\boldsymbol{d^\star}, \boldsymbol{b^\star})$ *and reduces the objective.*

Lemma 7. *Consider any bike-optimal solution* $(\boldsymbol{d}, \boldsymbol{b})$ *and a better allocation* $(\boldsymbol{d^\star}, \boldsymbol{b^\star})$*; let* j *and* k *denote stations with* $d_j + b_j < d^\star_j + b^\star_j$ *and* $d_k + b_k > d^\star_k + b^\star_k$*. Then either there exist* $(\boldsymbol{d'}, \boldsymbol{b'}) \in N(\boldsymbol{d}, \boldsymbol{b})$ *with* $d'_j + b'_j = d_j + b_j + 1$ *and* $(\boldsymbol{d^{\star\star}}, \boldsymbol{b^{\star\star}}) \in N(\boldsymbol{d^\star}, \boldsymbol{b^\star})$ *with* $d^{\star\star}_j + b^{\star\star}_j = d^\star_j + b^\star_j - 1$ *or there exist* $(\boldsymbol{d'}, \boldsymbol{b'}) \in N(\boldsymbol{d}, \boldsymbol{b})$ *with* $d'_k + b'_k = d_k + b_k - 1$ *and* $(\boldsymbol{d^{\star\star}}, \boldsymbol{b^{\star\star}}) \in N(\boldsymbol{d^\star}, \boldsymbol{b^\star})$ *with* $d^{\star\star}_k + b^{\star\star}_k = d^\star_k + b^\star_k + 1$ *such that*

1. $c(\boldsymbol{d}, \boldsymbol{b}) - c(\boldsymbol{d'}, \boldsymbol{b'}) \geq c(\boldsymbol{d^{\star\star}}, \boldsymbol{b^{\star\star}}) - c(\boldsymbol{d^\star}, \boldsymbol{b^\star})$
2. *the dock-move distance from* $(\boldsymbol{d'}, \boldsymbol{b'})$ *to* $(\boldsymbol{d^\star}, \boldsymbol{b^\star})$ *is one less than from* $(\boldsymbol{d}, \boldsymbol{b})$ *and the dock-move distance from* $(\boldsymbol{d^{\star\star}}, \boldsymbol{b^{\star\star}})$ *to* $(\boldsymbol{d}, \boldsymbol{b})$ *is one less than from* $(\boldsymbol{d^\star}, \boldsymbol{b^\star})$

Remark: In the discrete convexity literature, a rewriting of the objective allows this to be interpreted as the exchange property of $M^\natural$-convex functions; this connection has been explored in the follow-up work of Shioura [2018].

3.3 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 $\min\{D + B, z\}$ dock-moves apart. We begin by formally defining the set of feasible solutions w.r.t. the operational constraints.

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

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

Our proof works inductively. We begin by showing (Lemma 9) that, assuming that the solution in the zth iteration minimizes the objective among solutions at dock-move distance at most z, the solution found by the algorithm in the $z + 1$st iteration is a local optimum among solutions within $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$. In an earlier manuscript, as well as the proceedings version of the paper, we had falsely stated that local optima within $S_z(\boldsymbol{d}, \boldsymbol{b})$ are globally optimal within $S_z(\boldsymbol{d}, \boldsymbol{b})$, i.e., that Lemma 6 continues to hold in the setting where moves outside of $S_z(\boldsymbol{d}, \boldsymbol{b})$ are infeasible, and immediately derived optimality from that and Lemma 9. An example by Shioura [2018] shows that this is false: there exist solutions that are locally optimal with respect to our neighborhood structure and $S_z(\boldsymbol{d}, \boldsymbol{b})$, despite not being the best solution in $S_z(\boldsymbol{d}, \boldsymbol{b})$. (Shioura [2018] also provides an alternative proof of correctness for our algorithm). Nevertheless, we can use Lemma 9 to show the optimality of our algorithm: in Lemma 10 we show that the local optimality of the solution in the $z + 1$st iteration, with respect to $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$, guarantees a structural property. Then, in Theorem 11, we show that this structural property, together with the optimality of the solution in the zth iteration and the gradient-descent step, guarantee that the solution found by our algorithm in iteration $z+1$ is globally optimal among solutions in $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$.

Lemma 9. *Suppose* $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ *minimizes the objective among solutions in* $S_z(\boldsymbol{d}, \boldsymbol{b})$ *and let* $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ *denote the next choice of the gradient-descent algorithm, i.e., an allocation in the neighborhood of* $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ *that minimizes the objective. Then* $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ *is a local optimum within* $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$*, that is, there is no solution in* $N(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1}) \cap S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$ *with a lower objective.*

Proof. We know by Lemma 5 that $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ 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. If the $z+1$st move involved a third station then we denote it by h. (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 ℓ, including possibly $\ell = j$ or $\ell = h$, can be excluded since the greedy algorithm could have chosen to take a dock from ℓ instead of i and found a bike-optimal allocation (by Lemma 5).

2. The same holds for any dock-move in which a dock is taken from j.
3. A dock-move not involving any 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 $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$, then prior to the $z+1$st iteration it would have yielded a solution within $S_z(\boldsymbol{d}, \boldsymbol{b})$. Hence, by the induction assumption, it cannot yield any improvement.
4. A dock-move from station i (or to j), as is implied by inequalities (4), (5), and (6) 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. We discuss these in Appendix A.4.

Lemma 10. *Let $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ be defined as before and suppose there exists an allocation $(\boldsymbol{d}^\star, \boldsymbol{b}^\star) \in S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$ with smaller objective; then there exist j and k with $d_j^\star + b_j^\star > d_j^z + b_j^z \geq d_j + b_j$ and $d_k^\star + b_k^\star < d_k^z + b_k^z \leq d_k + b_k$.*

Proof. We know from Lemma 6 that there exists a solution in the neighborhood of $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ that has smaller objective than $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$, yet by Lemma 9 $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ is a local optimum within $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$; thus, such solutions must be outside of $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$. Consider in any such solution $(\boldsymbol{d}', \boldsymbol{b}')$ the station j that receives a dock, that is, $d_j^{z+1} + b_j^{z+1} + 1 = d_j' + b_j'$. Since $(\boldsymbol{d}', \boldsymbol{b}') \notin S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$, it must be the case that $d_j' + b_j' > d_j + b_j$ and thus $d_j^{z+1} + b_j^{z+1} \geq d_j + b_j$. What remains to be shown is that $d_j^z + b_j^z \geq d_j + b_j$; this is guaranteed since $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ has lower objective than $(\boldsymbol{d}^z, \boldsymbol{b}^z)$, is at larger dock-move distance from $(\boldsymbol{d}, \boldsymbol{b})$ than $(\boldsymbol{d}^z, \boldsymbol{b}^z)$, and is in the neighborhood of $(\boldsymbol{d}^z, \boldsymbol{b}^z)$. The proof for k is symmetric, based on the station k from which the dock is taken. □

Theorem 11. *Starting with a bike-optimal allocation $(\boldsymbol{d}, \boldsymbol{b})$, in the z-th iteration, the discrete gradient-descent algorithm finds an optimal allocation among those in $S_z(\boldsymbol{d}, \boldsymbol{b})$. Hence, the discrete gradient-descent algorithm terminates in at most $\min\{z, D+B\}$ iterations.*

Proof. Let $(\boldsymbol{d}^z, \boldsymbol{b}^z)$, $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ be defined as before and suppose $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$ is not the optimal allocation within $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$; we denote by $(\boldsymbol{d}^\star, \boldsymbol{b}^\star) \in S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$ an allocation that minimizes the distance to $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ among allocations in $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$ with smaller objective than $(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$. By Lemma 10 there exist stations j and k such that $d_j^\star + b_j^\star > d_j^z + b_j^z \geq d_j + b_j$ and $d_k^\star + b_k^\star < d_k^z + b_k^z \leq d_k + b_k$. Applying Lemma 7 to $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ and $(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$, we find that there exist $(\boldsymbol{d}', \boldsymbol{b}') \in N(\boldsymbol{d}^z, \boldsymbol{b}^z)$ and $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) \in N(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$ such that

$$c(\boldsymbol{d}^z, \boldsymbol{b}^z) - c(\boldsymbol{d}', \boldsymbol{b}') \geq c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) - c(\boldsymbol{d}^\star, \boldsymbol{b}^\star),$$

the dock-move distance from $(\boldsymbol{d}', \boldsymbol{b}')$ to $(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$ is one less than from $(\boldsymbol{d}, \boldsymbol{b})$, the dock-move distance from $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ to $(\boldsymbol{d}, \boldsymbol{b})$ is one less than from $(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$, and

either the dock-moves from $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ to $(\boldsymbol{d}', \boldsymbol{b}')$ and from $(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$ to $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ each involve j or they each involve k. Notice that $c(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1}) \leq c(\boldsymbol{d}', \boldsymbol{b}')$ by definition of the algorithm.

If $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) \in S_z(\boldsymbol{d}, \boldsymbol{b})$, then the assumption on $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ guarantees that $c(\boldsymbol{d}^z, \boldsymbol{b}^z) \leq c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$; in that case, the LHS can be bounded above by $c(\boldsymbol{d}^z, \boldsymbol{b}^z) - c(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$, whereas the RHS can be bounded below by $c(\boldsymbol{d}^z, \boldsymbol{b}^z) - c(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$, which contradicts $c(\boldsymbol{d}^\star, \boldsymbol{b}^\star) < c(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})$.

If instead $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) \in S_{z+1}(\boldsymbol{d}, \boldsymbol{b}) \setminus S_z(\boldsymbol{d}, \boldsymbol{b})$, then we have $(\boldsymbol{d}', \boldsymbol{b}') \in S_z(\boldsymbol{d}, \boldsymbol{b})$: since at station j, respectively k, the distance to $d_j + b_j$, respectively $d_k + b_k$, increases by 1 both from $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ to $(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$ and from $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ to $(\boldsymbol{d}', \boldsymbol{b}')$, at the station where the dock is taken to go from $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ to $(\boldsymbol{d}', \boldsymbol{b}')$, we must get one closer to $(\boldsymbol{d}, \boldsymbol{b})$. But then, the LHS can be bounded above by 0, and the RHS can be bounded below by $c(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1}) - c(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$, again yielding a contradiction. □

4 Scaling Algorithm

We now extend our analysis in Section 3 to adapt our algorithm to a scaling algorithm that provably finds an optimal allocation of bikes and docks, for the setting without dock movement constraints, in $O\big(n \log(B + D)\big)$ iterations. The idea underlying the scaling algorithm is to proceed in $\lfloor \log_2(B + D) \rfloor + 1$ phases, where in the kth phase each move involves $\alpha_k = 2^{\lfloor \log_2(B+D) \rfloor + 1 - k}$ bikes/docks rather than just one. The kth phase is prefaced by finding the bike-optimal allocation of bikes (given the constraints of only moving α_k bikes at a time), and terminates when no move of α_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 section implies that in the kth phase, the scaling algorithm finds the optimal allocation among all that differ in a multiple of α_k in each coordinate from $(\bar{\boldsymbol{d}}, \bar{\boldsymbol{b}})$. Further, since $\alpha_{\lfloor \log_2(B+D) \rfloor + 1} = 1$, it finds the globally optimal allocation in phase $\lfloor \log_2(B + D) \rfloor + 1$. What remains to be shown is a bound on the number of iterations in each phase.

Lemma 12. *The number of iterations in each phase is bounded by $O(n)$.*

Theorem 13. *The described scaling algorithm finds an optimal allocation for the problem without dock movement constraints in $O\big(n \log(B + D)\big)$ iterations.*

We provide the proof of Lemma 12 in Appendix A.5 and thereby prove Theorem 13. We then also extend the scaling algorithm to the problem with dock-movement constraints and show it can be solved in polynomial time; an even faster algorithm has been suggested by Shioura [2018].

5 Case Studies

In this section we present the results of case studies based on data from three different bike-sharing systems: Citi Bike in NYC, Hubway in Boston (recently

rebranded as Blue Bikes), and Divvy in Chicago. Some of our results are based on an extension of the user dissatisfaction function which we first define in Section 5.1. Thereafter, in Section 5.2 we describe the data-sets underlying our computation. Finally, in Section 5.3 we describe the insights obtained from our analysis.

5.1 Long-Run-Average Cost

A topic that has come up repeatedly 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 at all happens overnight. As such, it is desirable to optimize for reallocations that are robust with respect to the amount of overnight rebalancing. To capture such an objective, we define the long-run average of the user dissatisfaction function. Rather than mapping an initial condition in bikes and empty docks to the expected number of out-of-stock events over the course of one day, the long-run average maps to the average number of out-of-stock events over the course of infinitely many days; for that regime, the allocation of bikes is irrelevant. Formally, denoting by $X \oplus Y$ the concatenation of arrival sequences X and Y, i.e., $(X_1, \dots, X_t, Y_1, \dots, Y_s)$, we define the long-run average of a station i with demand profile p_i as follows.

Definition 14. *The long-run-average of the user dissatisfaction function at station i with demand profile p_i is*

$$c_i^\pi(d,b) = \lim_{T\to\infty} \frac{\mathbb{E}_{Y_j \sim p_i (i.i.d.)}[c^{Y_1 \oplus Y_2 \oplus \dots \oplus Y_T}(d,b)]}{T}.$$

We can compute $c^\pi(d,b)$ by computing for a given demand profile p_i 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, and given that each sequence of arrivals X occurs with probability $p_i(X)$. Given the resulting transition probabilities, we define a discrete Markov chain on $\{0, \dots, d_i + b_i\}$ and denote by $\pi_{p_i}^{d_i+b_i}$ its stationary distribution. This permits us to compute $c^\pi(d,b) = \sum_{k=0}^{d+b} \pi_{p_i}^{d+b}(k) c_i(d+b-k,k)$. Furthermore, from the definition of $c^\pi(\cdot,\cdot)$ it is immediately clear that $c^\pi(\cdot,\cdot)$ is also multimodular; as such all results proven in the previous sections about $c(\cdot,\cdot)$ also extend to $c^\pi(\cdot,\cdot)$. In addition, we observe that $c^\pi(\cdot,\cdot)$ depends only on the sum of its two arguments but not on the value of each (as the initial number of bikes does not influence the steady-state number of bikes). Before comparing the results of optimizing over $c^\pi(\cdot,\cdot)$ and over $c(\cdot,\cdot)$, we now give some intuition for why the long-run average provides a contrasting regime.

Intuition for the Long-run Average. It is instructive to consider examples to illustrate where optimizing over the long-run average deviates from optimizing over a single day. To simplify matters, we restrict ourselves to deterministic

demand profiles. A station at which the sequence of arrivals consists of k rentals followed by k returns has the long-run average of its user dissatisfaction decrease by 2 for each of the first k docks allocated; similarly, the user dissatisfaction function over a single day decreases by 2 for each full dock added (and by 1 for each empty dock added). At a station at which only k rentals occur, the user dissatisfaction function also decreases by 1 for each of the first k full docks added; however, its long-run average remains unchanged: no matter how many docks and bikes are added, the long-run average of the station is to be empty at the beginning of the day and therefore all k customers experience out-of-stock events.

Two lessons can be derived from these examples. First, stations at which demand is antipodal (rentals in the morning, returns in the afternoon or vice-versa) tend to make better use of additional capacity in the long-run average regime. Second, optimizing over one regime can, in theory, return solutions that are very bad in the other.

5.2 Data Sets

We use data-sets from the bike-sharing systems of three major American cities to investigate the effect different allocations of docks might have in each city. The three cities, New York City, Boston, and Chicago, vary widely in the sizes of their systems. When the data was collected from each system's open data feed (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. Given that the feeds only provide the number of bikes in each station, they 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.

For each station (in each system), we compute piece-wise constant Poisson arrival rates to inform our demand profiles. To be precise, we take all weekday rentals/returns in the month of June 2016, bucket them in the 30-minute interval of the day in which they occur, and divide the number of rentals/returns at each station within each half-hour interval by the number of minutes during which the station was non-empty/non-full. We compute the user dissatisfaction functions assuming that the demand profiles stem from these Poisson arrivals (cf. O'Mahony et al. 2016 and Parikh and Ukkusuri 2014). Some of our results in this section rely on the same procedure with data collected from other months.

Given that (in practice) we do not usually know the lower and upper bounds on the size of each station, we set the lower bound to be the current minimum capacity within the system and the upper bound to be the maximum one. Furthermore, we assume that $D + B$ is equal to the current allocated capacity in the system, i.e., we only reallocate existing docks.

5.3 Impact on Objective.

We summarize our results in Table 1. The columns Present, OPT, and 150-moved compare the objective with (i) the allocation before any docks are moved, (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 columns headed c contain the bike-optimal objective for a given allocation of docks, the columns headed c^π the long-run-average objective (for the same dock allocation). Two interesting observations can be made. First, though the optimizations are done over bike-optimal allocations without regard to the long-run average, the latter improves significantly in all cases. Second, in each of the cities, moving 150 docks yields a significant portion of the total possible improvement. 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 diminishing returns of the moves.

	Present		OPT		150-moved		Moves to OPT
City	c	c^π	c	c^π	c	c^π	
Boston	854	1118	640	943	700	984	407
Chicago	1460	2340	759	1846	1224	2123	1553
NYC	6416	9475	4829	8180	6150	9192	2721

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

A more complete picture of these insights is given in Figure 5. The x-axis shows the number of docks moved starting from the present allocation, the y-axis shows the cumulative *improvement in objective*, i.e., the difference between the initial objective and the objective after moving x docks. Each of the solid lines corresponds to different demand estimates being used to evaluate the same allocation of docks. The dotted lines (in the same colors) represent the maximum improvement, for each of the demand estimates, that can be achieved by reallocating docks; while these are not achieved through the dock moves suggested by the estimates based on June 2016 data, 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 11.

Seasonal Effects. As we mentioned in Section 2 we also consider the impact of seasonal effects. In Table 2 we show the improvement in objective when optimizing the movement of 200 docks in New York City based on demand estimates in June 2016 and evaluate the objective with the long-run average based on demand

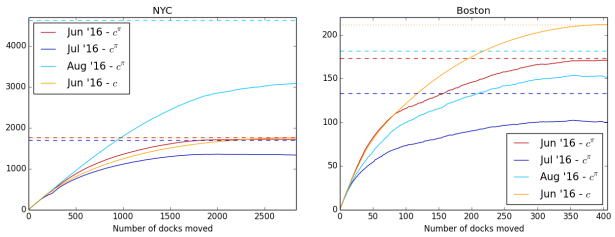

Fig. 5. Improvement in objective for moves to bike-optimal allocation for June '16 data.

estimates based on March and November 2017. The estimated improvements suggest that optimizing with respect to June yields significant improvement with respect to any other.

	June 2016	March 2017	November 2017
New York City	358.7	260.3	294.6

Table 2. Improvement of 200 docks moved based on long-run average evaluated with demand estimates June 2016, evaluated with demand estimates from 2017.

Operational Considerations. 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]). A simple coupling argument implies that a single bike yields at most a change of 1 in the user dissatisfaction function; thus, rebalancing reduced out-of-stock events by *at most* 3,452 per day (assuming that each rebalanced bike actually has that much impact is extremely optimistic). Contrasting that to the estimated impact of strategically moving, for example, 500 docks diminishes the estimated number of out-of-stock events *by about as much as a fifth* of Citi Bike's (daily) rebalancing efforts.

Second, discussions with operators uncovered an additional operational constraint that can arise due to the physical design of the docks. Since these usually 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 almost all of the improvements can be realized.

6 A Posteriori Evaluation of Impact

In this section, we apply the user dissatisfaction function to estimate the impact implemented changes in the system have had on out-of-stock events. One way to do so would be to estimate new demand rates after docks have been reallocated, compute new user dissatisfaction functions for stations with added (decreased) capacity, and evaluate for those stations and the new demand rates the decrease (increase) between the old and the new number of docks. A drawback of such an approach is the heavy reliance on the assumed underlying stochastic process. Instead, we present here a data-driven approach with only little reliance on assumed underlying demand profiles.

Throughout this section, we denote by d and b the number of empty docks and bikes at a station after docks were reallocated, whereas d' and b' denote the respective numbers before docks were reallocated. Notice that while $d + b$ and $d' + b'$ are known (capacity before and after docks were moved) and b can be found on any given morning (number of bikes in the station at 6AM), we rely on some assumed value for b' — for that, in our implementation, we picked both $\min\{d' + b', b\}$ and $b \times \frac{(d'+b')}{d+b}$, that is, either the same number of bikes (unless that would be larger than the old capacity before docks were added) or the same proportion of docks filled with bikes.

6.1 Arrivals at Stations with Increased Capacity

In earlier sections, we assumed a known distribution for the sequence of arrivals based on which we compute the user dissatisfaction functions. In contrast, in this section we rely exclusively on observed arrivals (without any assumed knowledge of the underlying stochastic process) to analyze stations with increased capacity. This is motivated by a coupling argument to justify that censoring need not be taken care of explicitly in this case. To formalize our argument, we need to introduce some additional notation for the arrival sequences. Recall from Section 2 that we denoted by $X = (X_1, \ldots, X_S)$ a sequence of customers arriving at a bike-share station to either rent or return a bike and that X included failed rentals and returns, which in practice would not be observed because they are censored. Which X_i are censored depends on the (initial) number of bikes and docks at the station. Let us denote by $X(d, b)$ the subsequence of X that only includes those customers whose rentals/returns are successful (hence, non-censored) at a station that is initialized with d empty docks and b bikes, i.e., the ones who do not experience out-of-stock events. Given the notation $c^X(\cdot, \cdot)$ used in Section 2 for a particular sequence of arrivals, we can then compute $c^{X(d,b)}(\cdot, \cdot)$. In particular, denoting by d', b' the number of empty docks and bikes without the added capacity, we may compute $c^{X(d,b)}(d', b')$. The following proposition then motivates the notion that censoring may be ignored at stations with added capacity.

Proposition 15. *For any X, $d' \leq d$ and $b' \leq b$, we have*

$$c^X(d', b') - c^X(d, b) = c^{X(d,b)}(d', b') - c^{X(d,b)}(d, b) = c^{X(d,b)}(d', b').$$

Proof. The proof of the second equality follows immediately from $X(d, b)$ including exactly those customers among X that are not censored, when a station is initialized with d empty docks and b bikes, so $c^{X(d,b)}(d, b) = 0$. Now, on the left-hand side, we can inductively go through all customers among X that are out-of-stock events when the station is initialized with d empty docks and b bikes. Since $d \geq d'$ and $b \geq b'$, each one of those increases both terms in the difference by 1. Thus, taking them out of X does not affect the value of the difference. But then, we are left with only $X(d, b)$. □

Extension to Stations with Decreased Capacity. Proposition 15 does not apply to stations with decreased capacity: suppose $d < d'$ and $b = b'$; once the station (initialized with d empty docks and b bikes) becomes full, $X(d, b)$ observes no further returns even though these would be part of $X(d', b')$. To account for out-of-stock events occurring in that way, we fill in the censored periods with demand estimates. This does not usually require knowledge of the full demand-profile; for example, for a station that is non-empty and non-full over the course of the day, no estimates are needed at all. Further, for periods of time in which the station is full, we only need to estimate the number of intended returns – rentals over that period of time would not be censored.

Extension to Rebalancing Based on our reasoning in Section 2.1, our analysis of the user dissatisfaction functions and the resulting optimization problem (cf. Sections 2 and 3) so far did not consider the rebalancing of bikes. In contrast, in the a posteriori analysis, we are able to take rebalancing into account.

To simplify the exposition, we restrict ourselves here to rebalancing that adds bikes to a station, though the reasoning extends to rebalancing that removes bikes. The simplest approach to treat bikes added through rebalancing is to just treat them as returns and thus include them (as virtual customers) in the sequence of arrivals X. However, this may cause an unreasonable increase to the value of $c^{X(d,b)}(d', b')$ (when the number of bikes added is greater than the number of empty docks would have been at that point in time if the station had initially had d' empty docks). In that case, the virtual customers (corresponding to rebalanced bikes) would incur out-of-stock events and thereby increase the value of the user dissatisfaction function. A more optimistic method that also treats rebalanced bikes as virtual customers would be to redefine the user dissatisfaction function in such a way, that out-of-stock events are only incurred by returns that correspond to non-rebalanced bikes. This, in essence, decouples the user dissatisfaction functions into subsequences, each of which are evaluated independently.

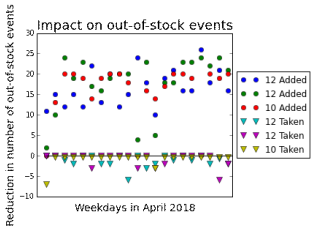

Fig. 6. Evaluation of impact at stations with increased and decreased capacity.

6.2 Impact of Initial Dock Moves in NYC

We consider 3 stations at which capacity was increased and 3 stations at which it was decreased based on our recommendations. For two of the stations at which capacity was increased, 12 docks were added, for one of them the capacity was increased by 10; the decreases were by the same amounts, so in total this involved reallocating 34 docks. In Figure 6 we present the estimated impact for each weekday in April 2018 (without the extension to rebalancing). For stations with added capacity we set d and b according to the number of bikes at 6AM. We evaluated $c^{X(d,b)}(d', b')$ for stations with docks added (cf. Proposition 15) using the observed arrivals $X(d, b)$ for each day. For the stations with docks taken away we estimated X by assuming a Poisson number of rentals (returns) whenever the station was empty (full), where the rate is based on decensored estimated demand from the same month. We use that to compute $c^{X(d,b)}(d', b') - c^{X(d,b)}(d, b)$ for these stations. The resulting values for different implementations are summarized in Table 3; aggregated over the entire month, the net reduction in out-of-stock events varies between 831 and 1062.

	No Rebalancing		Rebalancing	
	$\min\{b, d' + b'\}$	$b \times \left(\frac{d'+b'}{d+b}\right)$	$\min\{b, d' + b'\}$	$b \times \left(\frac{d'+b'}{d+b}\right)$
Decrease where capacity was added	831.0	1121.0	882.0	1027.0
Increase where capacity was taken	0	58.7	0	59.7
Net Reduction	831.0	1062.3	882.0	967.3

Table 3. Estimated cumulative changes at stations affected by dock reallocations based on the different evaluations described in Section 6.1.

7 Conclusion

We have considered several models that capture central questions in the design of dock-based bike-sharing systems, as are currently prevalent in North America. These models gave rise to new algorithmic discrete optimization questions,

and we have demonstrated that they have sufficient mathematical structure to permit their efficient solution, thereby also extending existing theory in discrete convexity. We have focused on the (re-)allocation of docks throughout the footprint of a bike-sharing system, capturing aspects of both better positioning of existing docks, and the optimal augmentation of an existing system with additional docks. These algorithms and models have been employed by systems within the United States with the desired effect of improving their day-to-day performance.

An alternative to optimizing dock allocation is to abandon the need to do so at all, by means of adopting a so-called dockless system. This approach has become prevalent in China, and is gradually being implemented in North America on a much smaller scale (both in comparison to the systems in China, and to the dock-based systems in North America); the management of these systems has its own challenges, and it remains to be seen whether these challenges can be overcome. Hybrid systems in which differential pricing enables centralized docking/parking areas that work in concert with dockless bikes may provide another path forward, as is done, for example, in Portland's Biketown system. Extensions of the methods we developed here will likely see continued use in this new setting as well.

Acknowledgment The authors gratefully acknowledge the collaboration with Motivate (the company operating Citi Bike), partial funding through NSF grants CCF-1526067, CMMI-1537394, CCF-1522054, and CCF-1740822, and ARO grant W911NF-17-1-0094, and the comments of A. Shioura, who identified a false claim in an earlier version of the manuscript.

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.

Hangil Chung, Daniel Freund, and David B Shmoys. Bike angels: An analysis of citi bike's incentive program. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, page 5. ACM, 2018.

Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. Setting inventory levels in a bike sharing network. *Transportation Science*, 2017.

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, Shane G Henderson, and David B Shmoys. Minimizing multimodular functions and allocating capacity in bike-sharing systems. *arXiv preprint arXiv:1611.09304*, 2016a.

Daniel Freund, Ashkan Norouzi-Fard, Alice Paul, Shane G. Henderson, and David B. Shmoys. Data-driven rebalancing methods for bike-share systems. working paper, 2016b.

Daniel Freund, Shane G Henderson, and David B Shmoys. Minimizing multimodular functions and allocating capacity in bike-sharing systems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 186–198. Springer, 2017.

Satoru Fujishige and Kazuo Murota. Notes on l-/m-convex functions and the separation theorems. *Mathematical Programming*, 88(1):129–146, 2000.

Supriyo Ghosh, Michael Trick, and Pradeep Varakantham. Robust repositioning to counter unpredictable demand in bike sharing systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3096–3102. AAAI Press, 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*, 2015.

Mor Kaspi, Tal Raviv, and Michal Tzur. Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. *IISE Transactions*, 49(2):144–158, 2017. doi: 10.1080/0740817X.2016.1224960. URL `http://dx.doi.org/10.1080/0740817X.2016.1224960`.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

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. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2016.

Satoko Moriguchi, Kazuo Murota, Akihisa Tamura, and Fabio Tardella. Discrete midpoint convexity. *arXiv preprint arXiv:1708.04579*, 2017.

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.

Kazuo Murota. Note on multimodularity and l-convexity. *Mathematics of Operations Research*, 30(3):658–661, 2005.

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.

Eoin O'Mahony, Shane G. Henderson, and David B. Shmoys. (Citi)Bike sharing. working paper, 2016.

Pulkit Parikh and Satish Ukkusuri. Estimation of optimal inventory levels at stations of a bicycle sharing system. In *Transportation Research Board Annual Meeting*, 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 Hoeve. 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`.

Akiyoshi Shioura. M-convex function minimization under l1-distance constraint. *arXiv preprint arXiv:1809.03126*, 2018.

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 O'Mahony, David B Shmoys, and Dawn B Woodard. Predicting Bike Usage for New York City's Bike Sharing System. In *AAAI Workshop: Computational Sustainability*, 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.

Appendix A Omitted Proofs

A.1 Proof of Lemma 2

We prove the lemma by induction, showing that $c^{X(t)}(\cdot,\cdot)$ 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). 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 by simly considering $t=1$ as the start of time. 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 $t-1$ customers.

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

A.2 Proof of Lemma 5

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 $k \in \{0, \ldots, d+b\}$. Thus, if the best allocation out of $e_{ij}(\boldsymbol{d}, \boldsymbol{b}), o_{ij}(\boldsymbol{d}, \boldsymbol{b}), E_{ijh}(\boldsymbol{d}, \boldsymbol{b})$, and $O_{ijh}(\boldsymbol{d}, \boldsymbol{b})$, 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 $(\boldsymbol{d}, \boldsymbol{b})$, 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 $\{e_{ij}, o_{ij}, E_{ijh}, O_{i,j,h}\}_{i,j,h \in [n]}$ – the other three cases are almost symmetric. Let ℓ denote a generic third station. Then a bike move that improves the objective could correspond to one being moved from ℓ to j, from i to j, from i to ℓ, from ℓ to i, from j to ℓ or from j to i. In this case, moves from ℓ to j, i to j and i to ℓ yield the allocations $E_{ij\ell}(\boldsymbol{d}, \boldsymbol{b})$, $o_{ij}(\boldsymbol{d}, \boldsymbol{b})$ and $O_{ij\ell}(\boldsymbol{d}, \boldsymbol{b})$, respectively. Since e_{ij} is assumed to be the minimizer among the possible dock-moves, none of these have objective smaller than that of $e_{ij}(\boldsymbol{d}, \boldsymbol{b})$. It remains to show that moving a bike from ℓ to i, j to ℓ or j to i yields no improvement. These all follow from bike-optimality of $(\boldsymbol{d}, \boldsymbol{b})$ and the multimodular inequalities. Specifically, an additional bike at i yields less improvement and one bike fewer at j has greater cost in $e_{ij}(\boldsymbol{d}, \boldsymbol{b})$ than in $(\boldsymbol{d}, \boldsymbol{b})$, since

$$c_i(d_i - 1, b_i) - c_i(d_i - 2, b_i + 1) \leq c_i(d_i, b_i) - c_i(d_i - 1, b_i + 1)$$
$$c_j(d_j + 2, b_j - 1) - c_j(d_j + 1, b_j) \geq c_j(d_j + 1, b_j - 1) - c_j(d_j, b_j).$$

Both of the above inequalities follow from inequality (3). □

A.3 Proofs of Lemmas 6 and 7

Proof of Lemma 6 We first argue that we may assume without loss of generality that there is no solution $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ that has a better objective than $(\boldsymbol{d}, \boldsymbol{b})$ and is one dock-move closer to $(\boldsymbol{d}, \boldsymbol{b})$ than is $(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$; if there was, we could induct on such a solution, as any dock move from $(\boldsymbol{d}, \boldsymbol{b})$ towards $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ would also be a move towards $(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$.

Now, the proof of the lemma follows a case-by-case analysis; since $(\boldsymbol{d}, \boldsymbol{b})$ is bike-optimal, there must be stations j and k such that $d_j + b_j < d_j^{\star} + b_j^{\star}$ and $d_k + b_k > d_k^{\star} + b_k^{\star}$. We show that in each case either there exists a dock-move to j, or one from k, that improves the objective or there exists a solution $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ with objective value lower than $(\boldsymbol{d}, \boldsymbol{b})$, $\sum_i d_i + b_i = \sum_i d_i^{\star\star} + b_i^{\star\star}$, and $\sum_i b_i = \sum_i b_i^{\star\star}$, such that $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ is one dock-move closer $(\boldsymbol{d}, \boldsymbol{b})$ than is $(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$. Given that we may assume without loss of generality that no such $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ exists, this proves that in $(\boldsymbol{d}, \boldsymbol{b})$ there must be a dock-move to j or one from k that yields a lower objective. We distinguish among the following cases:

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

(a) and there exists ℓ with $d_\ell + b_\ell > d^\star_\ell + b^\star_\ell$ and $b_\ell > b^\star_\ell$;
(b) and there exists ℓ with $d_\ell + b_\ell \leq d^\star_\ell + b^\star_\ell$ and $b_\ell > b^\star_\ell$;
(c) for all $\ell \notin \{j, k\}$, we have $b_\ell \leq b^\star_\ell$, so $\sum_i b_i < \sum_i b^\star_i$.

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 ℓ, respectively from ℓ to j, that yields improvement. Since the proofs for cases (3b) and (4b) are also symmetric, we only present the proof for (3b). Cases (3c) and (4c) contradict our assumption that $\sum_i b_i = \sum_i b^\star_i$ and can thus be excluded. For case (1), we define $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) = e_{jk}(\boldsymbol{d}^\star, \boldsymbol{b}^\star)$, so

$$c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) - c(\boldsymbol{d}^\star, \boldsymbol{b}^\star) = c_j(d^\star_j - 1, b^\star_j) - c_j(d^\star_j, b^\star_j) + c_k(d^\star_k + 1, b^\star_k) - c_k(d^\star_k, b^\star_k).$$

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

$$c(\boldsymbol{d}, \boldsymbol{b}) - c(\boldsymbol{d'}, \boldsymbol{b'}) = \Big(c_j(d_j, b_j) - c_j(d_j + 1, b_j)\Big) + \Big(c_k(d_k, b_k) - c_k(d_k - 1, b_k)\Big)$$
$$\geq \Big(c_j(d^\star_j - 1, b^\star_j) - c_j(d^\star_j, b^\star_j)\Big) + \Big(c_k(d^\star_k + 1, b^\star_k) - c_k(d^\star_k, b^\star_k)\Big) = c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) - c(\boldsymbol{d}^\star, \boldsymbol{b}^\star) > 0.$$

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

$$c_j(d_j, b_j) - c_j(d_j + 1, b_j) \geq c_j(d^\star_j - 1, b^\star_j) - c_j(d^\star_j, b^\star_j).$$

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

$$c_j(d_j, b^\star_j + d^\star_j - d_j - 1) - c_j(d_j + 1, b^\star_j + d^\star_j - d_j - 1).$$

On the other hand, applying inequality (6) repeatedly to the 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 $s = b^\star_j + d^\star_j - d_j - b_j - 1$, which is non-negative since $b_j + d_j < b^\star_j + d^\star_j$, we bound the LHS from below by

$$c_j(d_j, b_j + b^\star_j + d^\star_j - d_j - b_j - 1) - c_j(d_j + 1, b_j + b^\star_j + d^\star_j - d_j - b_j - 1).$$

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

$$c_k(d_k - 1, b_k) - c_k(d_k, b_k) \leq c_k(d^\star_k, b^\star_k) - c_k(d^\star_k + 1, b^\star_k), \tag{7}$$

we apply inequality (3) $d_k - d^\star_k - 1$ times to bound the LHS in (7) by $c_k(d^\star_k, b_k + d_k - d^\star_k + 1) - c_k(d^\star_k + 1, b_k + d_k - d^\star_k + 1)$. 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 $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) = E_{jk\ell}(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$ and $(\boldsymbol{d'}, \boldsymbol{b'}) = O_{kj\ell}(\boldsymbol{d}, \boldsymbol{b})$. Similarly to the first case, we need to show that $c(\boldsymbol{d}, \boldsymbol{b}) - c(\boldsymbol{d'}, \boldsymbol{b'}) \geq c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) - c(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$. Since all terms not involving j, k, and ℓ cancel out and the terms involving j and k can be bounded the same way as before, deriving

$$c_\ell(d_\ell, b_\ell) - c_\ell(d_\ell - 1, b_\ell + 1) \geq c_\ell(d^\star_\ell + 1, b^\star_\ell - 1) - c_\ell(d^\star_\ell, b^\star_\ell)$$

suffices. We obtain this by repeatedly applying inequalities (3) and (4) to the LHS. This concludes the proof of Lemma 6. □

Proof of Lemma 7 The constructions in the proof of Lemma 6 fulfill the requirements of Lemma 7; the only required change is due to the assumption that, without loss of generality, there is no solution $(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star})$ that has a better objective than $(\boldsymbol{d}, \boldsymbol{b})$ and is one dock-move closer to $(\boldsymbol{d}, \boldsymbol{b})$ than $(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star})$. However, that assumption is only ever used to derive the final inequality in

$$c(\boldsymbol{d}, \boldsymbol{b}) - c(\boldsymbol{d'}, \boldsymbol{b'}) \geq c(\boldsymbol{d}^{\star\star}, \boldsymbol{b}^{\star\star}) - c(\boldsymbol{d}^{\star}, \boldsymbol{b}^{\star}) > 0.$$

For the statement of Lemma 7 we only require the first inequality, which is thus guaranteed.

A.4 Remaining Cases in Proof of Lemma 9

The remaining cases are ones that involve 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 $E_{h\ell m}$, $o_{\ell h}$, and $E_{\ell h m}$ are similar. Notice first that if it were the case that $d^z_h + b^z_h > d_h + b_h$ and $d^z_\ell + b^z_\ell < d_\ell + b_\ell$, then $e_{h\ell}(E_{ijh}(\boldsymbol{d}^z, \boldsymbol{b}^z)) \in S_z(\boldsymbol{d}, \boldsymbol{b})$ and has a lower objective than $(\boldsymbol{d}^z, \boldsymbol{b}^z)$ which contradicts the inductive assumption. Furthermore, since it must be the case that $e_{h\ell}(E_{ijh}(\boldsymbol{d}^z, \boldsymbol{b}^z))$ is an element of $S_{z+1}(\boldsymbol{d}, \boldsymbol{b})$ but not of $S_z(\boldsymbol{d}, \boldsymbol{b})$, it must also follow that either

1. $d^z_h + b^z_h > d_h + b_h$ and $d^z_\ell + b^z_\ell \geq d_\ell + b_\ell$ or
2. $d^z_h + b^z_h \leq d_h + b_h$ and $d^z_\ell + b^z_\ell < d_\ell + b_\ell$,

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

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

In the first case, since $o_{h\ell}(\boldsymbol{d}^z, \boldsymbol{b}^z) \in S_z(\boldsymbol{d}, \boldsymbol{b})$, 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 ℓ 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 ℓ 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 $c(e_{h\ell}(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1})) - c(\boldsymbol{d}^{z+1}, \boldsymbol{b}^{z+1}) \geq 0$.

In the second case, since we know that $e_{i\ell}(\boldsymbol{d}^z, \boldsymbol{b}^z) \in S_z(\boldsymbol{d}, \boldsymbol{b})$, 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 $z+1$. The 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 ℓ 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 ℓ, 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, $e_{im}(\boldsymbol{d}^z, \boldsymbol{b}^z) \in S_z(\boldsymbol{d}, \boldsymbol{b})$, 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 ℓ 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 ℓ. Hence, the decrease in objective at h and at m is bounded above by the increase in objective at ℓ.

In the latter case, the inductive assumption implies that an increase in objective at ℓ 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 $o_{\ell j}(\boldsymbol{d}^z, \boldsymbol{b}^z) \in S_z(\boldsymbol{d}, \boldsymbol{b})$). That improvement however is at least as large as 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. □

A.5 Appendix to Section 4

Proof of Lemma 12 By Theorem 11, 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 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 $e_{i\ell}$ occur, then the moves e_{ij}, e_{ij}, and $e_{k\ell}$ yield the same changes, but involve e_{ij} twice. The same argument holds for the other kinds of moves. By Theorem 11, 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 derive a contradiction from $L > 5n$ and thus prove the lemma. □

Extension to operational constraints While the same ideas as in Lemma 12 still work in the presence of constraints on the number of docks moved, we need to be careful to ensure that in each phase we remain within $S_z(\boldsymbol{d}, \boldsymbol{b})$. We adapt the scaling algorithm in the following way to ensure this: rather than starting each new phase at the optimal solution found in the last phase, we move that solution closer to the current solution and initiate the discrete-gradient descent steps from that new solution. Suppose in the last iteration we found a solution

at distance z from the current allocation that is optimal when only allowing multiples of 2^k to be moved; in order to find the optimal solution when allowing multiples of 2^{k-1}, we require the existence of a number M such that (i) M is polynomial in n, (ii) the new starting point of the phase is in $S_{z-2^{k-1}M}(\boldsymbol{d}, \boldsymbol{b})$, and (iii) there exists an optimal solution within $S_z(\boldsymbol{d}, \boldsymbol{b})$, for steps involving 2^{k-1} docks/bikes each, that is at most M such steps away from the new starting point. By Theorem 11, running M gradient descent steps then finds an optimal solution within $S_z(\boldsymbol{d}, \boldsymbol{b})$ in the new phase.

Lemma 16. *Let $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ be the optimal solution found in the phase in which we move 2^k docks/bikes in each iteration and let $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$ be the optimal solution for the subsequent phase that is closest (in dock-move distance) to $(\boldsymbol{d}^k, \boldsymbol{b}^k)$. Let α denote that dock-move distance (counting the number of moves of 2^{k-1} docks/bikes each). We define $(\boldsymbol{d}', \boldsymbol{b}')$ by setting:*

- *if $d_i^k + b_i^k \geq \bar{d}_i + \bar{b}_i$, then $d'_i + b'_i = \max\{d_i^k + b_i^k - \alpha 2^{k-1}, \bar{d}_i + \bar{b}_i\}$*
- *if $d_i^k + b_i^k < \bar{d}_i + \bar{b}_i$, then $d'_i + b'_i = \min\{d_i^k + b_i^k + \alpha 2^{k-1}, \bar{d}_i + \bar{b}_i\}$.*

Then, as long as α is polynomial in n, $M = \frac{\sum_i |d'_i + b'_i - d_i^k + b_i^k|}{2^k}$ fulfills properties (i)-(iii) above.

Proof. Property (i) holds with $M \leq \alpha n$; (ii) holds because each of the M docks moved from $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ to $(\boldsymbol{d}', \boldsymbol{b}')$ decreases the move-distance from the current allocation. We need to argue that property (iii) holds as well by showing that the dock-move distance from $(\boldsymbol{d}', \boldsymbol{b}')$ to $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$ is at most M.

Notice first that there is no station i such that either

- $d'_i + b'_i < \bar{d}_i + \bar{b}_i$ and $d_i^{k-1} + b_i^{k-1} > \bar{d}_i + \bar{b}_i$ or
- $d'_i + b'_i > \bar{d}_i + \bar{b}_i$ and $d_i^{k-1} + b_i^{k-1} < \bar{d}_i + \bar{b}_i$.

In fact, if there was such a location i, then $d_i^{k-1} + b_i^{k-1}$ would have to be greater than α moves (of step-size 2^{k-1}) away from $d_i^k + b_i^k$, contradicting the distance being α. We deduce that

$$\sum_i |d_i + b_i - d'_i - b'_i| + |d'_i + b'_i - d_i^{k-1} - b_i^{k-1}| = \sum_i |d_i + b_i - d_i^{k-1} - b_i^{k-1}| \leq z.$$

Since $\sum_i |d_i + b_i - d'_i - b'_i| = z - M$, it follows that $\sum_i |d'_i + b'_i - d_i^{k-1} - b_i^{k-1}| \leq M$ which concludes the proof of the Lemma. □

By Lemma 12, $\alpha \in O(n)$ when there are no dock-move constraints. We next show that, a looser bound of $\alpha \in O(n^3)$ can be obtained when we do have dock-move constraints.

Lemma 17. *With $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ and $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$ defined as in the last lemma, the dock-move distance between the two, where each move involves 2^{k-1} docks/bikes, is bounded by $O(n^3)$.*

Proof. Similar to Lemma 12 we again use a pigeonhole argument, however, this time we bucket each individual dock-move, i.e., for each pair/triple $(i,j)/(i,j,h)$ we consider all possible moves. A loose bound on the number of moves before duplicates arise is $4n^3$ (without the bucketing arguments we used in Lemma 12). Recall that the dock-move distance to the current allocation is assumed to be the same for $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ and $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$. Suppose β among the moves in the path from $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ to $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$ appear once and do not increase, or even decrease the dock-move distance to the current allocation. Then there can be at most β moves that increase the dock-move distance to the current. Further, there can be no repeated moves that decrease the dock-move distance, as we can combine two iterations of such a move and two iterations of a move that decreases the dock-move distance, to find a feasible solution in the previous phase. By the same argument as in Lemma 12, that solution would contradict the optimality of $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ in the previous phase. Thus, we know that $\beta \leq 4n^3$ as otherwise duplicates would occur; but then, there can be no more than $8n^3$ moves in the path from $(\boldsymbol{d}^k, \boldsymbol{b}^k)$ to $(\boldsymbol{d}^{k-1}, \boldsymbol{b}^{k-1})$. □

In this section we assumed so far that in each phase we search for the optimal solution at the same distance from the current allocation as in the last phase; in particular, we assumed that the distance from the current allocation is the same for the optimal solution in consecutive phases. In practice, we may have optimal solutions $O(n)$ moves further from the current allocation (by a similar argument as in Lemma 12, e.g., if the bounds at each station do not allow us to add an additional 2^k docks, but do allow us to add an additional 2^{k-1} docks). In that case, we may first find an optimal solution with moves of size 2^{k-1} for the distance in the last phase, and then carry out an additional number of moves, linear in n, to find the solution for the actual distance constraint. Combining these ideas we obtain a polynomial-time algorithm for the optimization problem.

Theorem 18. *The described algorithm runs in polynomial time.*

Appendix B Connections to M-Convex Functions

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

Definition 19 (M-convex set). *A nonempty set of integer points $B \subseteq Z^{2n}$ is defined to be an M-convex set if it satisfies $\forall \boldsymbol{x}, \boldsymbol{y} \in B, i \in supp^+(\boldsymbol{x}-\boldsymbol{y}), \exists j \in supp^-(\boldsymbol{x}-\boldsymbol{y}) : \boldsymbol{x} - \boldsymbol{e}_i + \boldsymbol{e}_j \in B$.*

Definition 20 (M-convex function). *A function f is M-convex if for all $x, y \in dom(f), i \in supp^+(x-y), \exists j \in supp^-(x-y) : f(x) + f(y) \geq f(x - e_i + e_j) + f(y + e_i - e_j)$.*

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

Lemma 21. *Murota [2003] For an M-convex function f and $x \in dom(f)$ we have $f(x) \leq f(y)\ \forall y \in dom(f)$ if and only if $f(x) \leq f(x - e_i + e_j) \forall i, j$.*

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

1: Find a vector $x \in dom(f)$
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

As the following 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 22. Our example consists of three stations i, j, and k with demand-profiles:

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

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 $1 = c(\boldsymbol{d}^*, \boldsymbol{b}^*) < c(\boldsymbol{d}', \boldsymbol{b}') = \frac{3}{2}$. But then the statement of Lemma 21 with $y = (\boldsymbol{d}^*, \boldsymbol{b}^*)$ and $x = (\boldsymbol{d}, \boldsymbol{b})$ implies that, if c is M-convex then one of $c((\boldsymbol{d}'_{-i}, d'_i{+}1), c(\boldsymbol{d}', (\boldsymbol{b}'_{-i,-k}, b'_i{-}1, b'_k{+}1))$, or $c((\boldsymbol{d}'_{-i,-j}, d'_i{+}1, d'_j{-}1), \boldsymbol{b}')$ must be strictly smaller than $c(\boldsymbol{d}', \boldsymbol{b}')$. 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.

Appendix C Connections to Discrete Midpoint Convex Functions

In this appendix we show that the constrained optimization problem formulated in Section 2 is not multimodular. To do so, we apply an equivalence proven in Murota [2005] that characterizes a function f as multimodular if and only if there exists an $L^\natural$ convex function g such that $f(x_1, x_2, \ldots, x_n) = g(x_1, x_1 + x_2, \ldots, \sum_{i=1}^n x_i)$. While we do not state the explicit definition of $L^\natural$ convex functions here, it was shown by Fujishige and Murota [2000] that $L^\natural$ convex functions fulfill the following discrete midpoint convexity property.

Definition 23. *A function $g : \mathbb{Z}^n \to \mathbb{R}^n \cup \{+\infty\}$ is called discrete midpoint convex if*

$$g(x) + g(y) \geq g\big(\lceil \frac{x+y}{2} \rceil\big) + g\big(\lfloor \frac{x+y}{2} \rfloor\big).$$

Here, the floor and ceiling refer to component-wise floor and ceiling.

We now argue that the function g corresponding to our (constrained) objective c is not discrete midpoint convex. Consider the current allocation (cf. Section 2) $\bar{\boldsymbol{d}} = (0,1,0,1)$ and $\bar{\boldsymbol{b}} = (0,0,0,0)$. As all values for $\boldsymbol{b}$ are 0 throughout this construction, we do not restate it from now on. Suppose $z = 1$, that is, only one dock is allowed to be moved (and solutions moving more than one are infeasible and thus have value infinity). Then the vector $\boldsymbol{d} = (1,0,1,0)$ is not feasible given the constraint (as it would involve moving 2 docks). Now, if g was discrete midpoint convex, then the inequality would state that

$$\begin{aligned} &f(1,0,0,1) + f(0,1,1,0) = g(1,1,1,2) + g(0,1,2,2) \\ &\geq g(1,1,2,2) + g(0,1,1,2) = f(1,0,1,0) + f(0,1,0,1). \end{aligned}$$

However, both terms on the left-hand side are feasible whereas the first term on the right-hand side is not. Thus, the inequality does not hold, g is not discrete midpoint convex, and therefore f is not multimodular.

Appendix D Tradeoff between number of reallocated and new docks

In this appendix, we show that the discrete gradient-descent algorithm can be applied, with little overhead, to solve a variation of the optimization problem introduced in Section 2. In this variation, rather than having fixed budgets that capture the number of docks and the number of docks that may be reallocated, we consider a setting in which there is a joint budget on both. To do so we introduce two new parameters. The parameter k captures how much more expensive it is to acquire new docks rather than reallocate existing ones. The parameter M bounds the joint cost of reallocating existing and acquiring new docks. In the systems we have worked with, k is so large that the optimal solution would rarely ever acquire new docks. However, it is conceivable that in other systems k would be much smaller; thus, we provide in this appendix both the new optimization problem and an explanation of how the discrete gradient-descent algorithm can be applied to solve it optimally. We begin with the formulation; here, $\bar{D}$ is a new decision variable that captures the number of newly acquired docks and z, previously a parameter, becomes a decision variable.

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

For each fixed pair of values of z and $\bar{D}$, the discrete-gradient descent algorithm finds an optimal solution by the analysis in Section 3. Furthermore, it is easily observed that for each value of $\bar{D}$, it is optimal to set $z = M - k\bar{D}$. Hence, one way of finding an optimal solution would be to try all $\lfloor \frac{M}{k} \rfloor$ feasible values of $\bar{D}$ (and corresponding values of z) and solve optimally with the corresponding value of z.

A better algorithm to find the optimal solution is based on the following observation: by Theorem 11, the dock-move distance between the optimal allocation for $\bar{D}$ and $z = M - k\bar{D}$ on the one hand and the one for $\bar{D}$ and $z = M - k(\bar{D}+1)$ on the other is at most $\bar{D}$. Hence, we only need to bound the distance to an optimal solution that has an additional empty dock at its disposal. A corollary of the analysis in Section 3.3 is that the dock-move distance from an optimal solution for a given budget to an optimal solution with one additional dock available is bounded by 1.

The reasoning above implies that the gradient-descent algorithm with a minimal adaptation can be used to solve the optimization problem that includes a tradeoff between the cost of new docks and the cost of reallocating docks; however, in practice this tradeoff barely ever arises, since the relative cost of new inventory greatly outweighs that of reallocating existing industry.

Appendix E Running Time

Even though the reallocation of docks is a strategic question, the time to solve the associated optimization models is not irrelevant for practical considerations. Given the expensive computation of each user dissatisfaction value, an early approach to compute the LP-relaxation of the optimization problem took a weekend to solve (on a high-end laptop). This was due to the time required to set up the LP; once it was set up, it solved quickly. While this is acceptable for a one-off analysis, in practice system operators care about regularly running different analyses that include different demand patterns, different bounds on number of docks moved, and even different bounds on station sizes. Having a fast algorithm allows system operators to run the analysis without our support. We provided them with a Jupyter notebook (Kluyver et al. 2016) that includes the entire workflow, from estimating the demand profiles to computing the user dissatisfaction functions to running the optimization problem to creating map-based visualizations of the resulting solutions (cf. Figure 7) and does not rely on specialized optimization software like Gurobi or CPLEX. Crucially, this workflow happens in a matter of minutes rather than hours or days (cf. Table 4).

To conclude this numerical exploration, we now compare the measured running times of the greedy and the scaling algorithm. Given that the running-time of each algorithm is dominated by the computational effort to compute values of the user dissatisfaction functions (the effort for which grows as the cube of the capacity), we only computed values to which the respective algorithm needed access. In Figure 8 we plot the number of user dissatisfaction functions that are computed by each algorithm. In Chicago, the scaling algorithm created unnec-

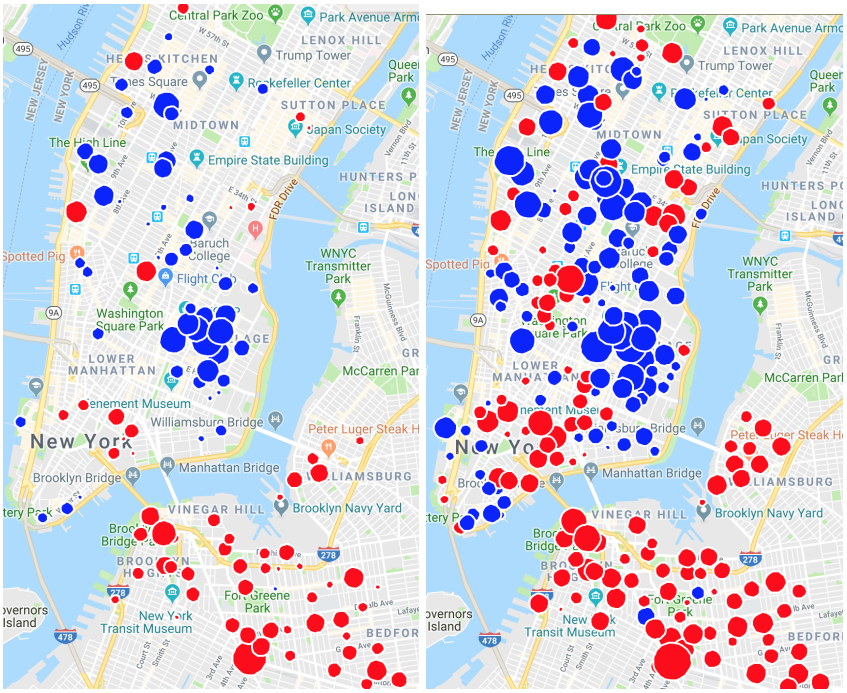

Fig. 7. Visualization of docks moved by optimal solution in NYC for $z \in \{500, 1500\}$; red circles correspond to stations at which docks are taken, blue circles to stations at which they are added.

essary overhead by requiring values 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 over 8, 4, and 1, rather than all powers of 2. The hybrid outperforms both the greedy and scaling algorithm on all three data-sets. All three algorithms vastly outperform the linear programming based approach that needs to evaluate every value of the user dissatisfaction functions at all stations before solving.

	Running Time (Minutes)		
	Greedy	Hybrid	Scaling
New York City	18.08	14.02	12.27
Chicago	7.03	5.67	8.78
Boston	1.44	1.37	1.83

Table 4. Comparison of the running times of each of the three algorithms in each of the three cities

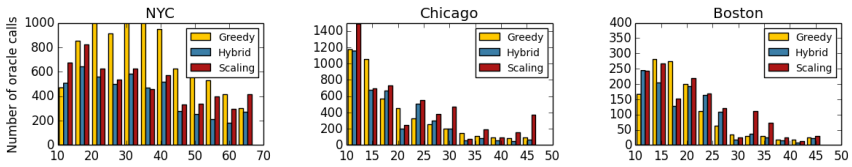

Fig. 8. Number of user dissatisfaction functions, grouped by capacity $d + b$, evaluated by each algorithm in each city.