Submitted to *Interfaces*
manuscript (Please, provide the manuscript number!)

Authors are encouraged to submit new papers to INFORMS journals by means of a style file template, which includes the journal title. However, use of a template does not certify that the paper has been accepted for publication in the named journal. INFORMS journal templates are for the exclusive purpose of submitting to an INFORMS journal and should not be used to distribute the papers in print or online or to submit the papers to another publication.

**Analytics and Bikes:**
Riding Tandem with Motivate to Improve Mobility

Daniel Freund  
Marketplace Labs, Lyft Inc., dfreund@lyft.com

Shane G. Henderson  
School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850, sgh9@cornell.edu

Eoin O’Mahony  
Uber Technologies Inc., eoin@uber.com

David B. Shmoys  
School of Operations Research & Information Engineering and Department of Computer Science, Cornell University, Ithaca, NY 14850, dbs10@cornell.edu

Bike-sharing systems are now ubiquitous across the United States. We have worked with Motivate, the operator of the systems in, for example, New York, Chicago, and San Francisco, to innovate a data-driven approach to managing both their day-to-day operations and to provide insight on several central issues in the design of its systems. This work required the development of a number of new optimization models, characterizing their mathematical structure, and using this insight in designing algorithms to solve them. Here, we focus on two particularly high impact projects, an initiative to improve the allocation of docks to stations, and the creation of an incentive scheme to crowdsource rebalancing. Both of these projects have been fully implemented to improve the performance of Motivate’s systems across the country; for example, the Bike Angels program in New York City yields a system-wide improvement comparable to that obtained through Motivate’s traditional rebalancing efforts, at far less financial and environmental cost.

*Key words:* Transportation, Optimization, Inventory, Sharing Economy
Introduction

Bike-sharing systems are now ubiquitous across the United States. Our industry partner, Motivate, operates the largest such systems: Citi Bike in New York City (NYC), Divvy in Chicago, Blue Bikes in Boston, and Ford GoBike in the Bay Area. Together, these systems accounted for more than 70% of the 35 million bike-sharing trips that occurred in the United States in 2017 (NACTO Bike Share Initiative 2018); since 2010 the annual number of bike-sharing trips in the United States has grown a hundredfold.

Motivate’s bike-sharing systems consist of a number of stations placed densely within each city; every station has a number of docks, each of which either holds a bike (is full) or is empty. In allowing users to rent a bike from any station with at least one bike and return it to any station with at least one empty dock, the systems provide commuters and tourists alike a sustainable transportation option. Yet, their success in increasing ridership goes hand in hand with a significant struggle to handle asymmetric demand due to “tidal” commuter flows. In bike-sharing, this asymmetry causes out-of-stock events, where users are unable to rent bikes at stations at which all docks are empty, or arguably even worse, where users are unable to return a bike at stations where all docks are full. Attempting to alleviate these effects of asymmetric demand, operators try to rebalance the system by moving bikes from full to empty stations, typically by box truck or van. For operators, this constitutes one of the largest operational expenses.

Contribution

“The work with Cornell was, in my mind, really about giving us an analytical foundation to be able to make some of the decisions that were most critical to what we are doing.”, Jay Walder, CEO, Motivate (2018).

In this paper, we describe how our work with Motivate has helped make the company an industry leader in the development and use of analytics to drive strategic, tactical and
operational decisions in bike-sharing systems. Our data-driven approach has informed the deployment of many operational levers to improve system performance. Beyond helping Motivate’s operations on a tactical level, our methodology has also informed its strategic vision for how to tackle imbalance. We describe two projects in which our work introduced new elements to Motivate’s overall system design that reduce the need for (motorized) rebalancing and its associated financial and environmental costs. Essentially, not only did we help Motivate rebalance more efficiently, but we also guided Motivate towards system designs that require less rebalancing in the first place. The first project formulated an optimization model to inform the allocation of docks across the system and, moreover, characterized its mathematical structure, thereby enabling the development of computationally efficient algorithmic tools for its solution. The second project set up an incentive program, prominently advertised as Bike Angels, to crowdsourcing rebalancing through the user base. Both of these projects have been fully implemented, and Motivate’s systems, and its users, benefit from these advances. The data-driven approach underlying these two projects has also been instrumental in driving operational decisions in Motivate’s rebalancing operations, but we do not describe in detail those innovations here. See Freund (2018) for a full account of these innovations.

This high-impact work has been enabled through innovations in stochastic modeling; in data analysis to fit stochastic models; in optimization formulations, built on the stochastic models that capture key characteristics of strategic, tactical, and operational considerations; and in providing sufficient mathematical understanding of these formulations so as to facilitate the design and analysis of highly efficient optimization algorithms for solving them. This paper summarizes these contributions and details their impact.

The foundation of our work with Motivate is the notion of user dissatisfaction functions (UDFs), as first defined by Raviv and Kolka (2013). These functions map a station’s
capacity and inventory level at the beginning of a planning period to the expected number of out-of-stock events over the course of the period. Computing UDFs requires selecting appropriate stochastic processes to model sequences of users wishing to rent and return bikes at stations, and, moreover, estimating the parameters of those stochastic processes. Demand data is, unfortunately, censored when stations are empty or full, due to out-of-stock events. We developed and employed decensoring techniques to obtain time-dependent demand rates for both arrivals and returns at each location (O’Mahony and Shmoys 2015). Further, we developed a novel, numerically stable method (O’Mahony 2015) to compute UDFs, based on the so-called Poisson equation for continuous-time Markov chains.

For the first project of allocating dock capacity, we provide a nonlinear integer program (IP) to optimize over UDFs with flexible capacity at each location. Discrete convexity properties, especially multimodularity as discussed later, that we establish using sample-path arguments then enable the development of an efficient algorithm that not only solves the IP optimally, but also furthers our understanding of constrained optimization in the setting of discrete convexity (Freund et al. 2017). Our algorithm works as follows: We start from an initial assignment of docks to stations, and first identify the optimal allocation of bikes to stations through a simple naturally integer linear program. Then, in each iteration, we determine the reallocation of at most one bike and at most one dock that most improves the objective function. This continues until no such improving move exists. The proof of correctness relies on two elements: first showing that for any realization of demand, the UDF at each station has a property known as multimodularity, and second showing that this structure is sufficient to guarantee that the algorithm always finds an optimal solution to the system-wide allocation problem. Based on solutions to the IP, Motivate carried out a pilot in NYC to reallocate dock capacity; we use subsequently collected usage data to
provide a UDF-based method that estimates the pilot’s impact in reducing the need to rebalance. The pilot’s estimated impact, along with the powerful tools we developed, have induced Motivate to embrace the principle of moving dock capacity, with hundreds of docks moved to date in its systems nationwide.

For the second project of designing an incentive scheme, the goal is to reward customers for rides that drive the system towards desirable configurations, thereby incentivizing customers to align their rides with desirable outcomes. But which rides should be incentivized? Again, UDFs play a central role. We explain how UDFs are employed, and discuss the marginal benefits of making the incentives dynamic in time, in response to real-time demand, in contrast to static policies that set them in advance.

Related Work

Given the growth in bike-sharing systems it is unsurprising that the study of their operations has become a popular area within the field of OR. We refer the reader to extensive literature reviews in Freund et al. (2018), Freund (2018), and de Chardon et al. (2016). Most important for our work are the contributions of Raviv and Kolka (2013), Schuijbroek et al. (2017), and Parikh and Ukkusuri (2014). Preliminary versions of our own contributions described here have previously appeared as O’Mahony and Shmoys (2015), O’Mahony et al. (2016), Freund et al. (2017), and Chung et al. (2018).

Imbalance, Rebalancing, and User Dissatisfaction Functions

“You’re better than this @DivvyBikes. Two stations completely full in West Town/Fulton Market. First time I’ve had to seek out a third.”, Ross (2017)

“@CitiBikeNYC Why are your guys removing bikes from Madison square park at 2:30? People will need them to get home at 5”, McCloskey (2015)

The efficient operation of bike-sharing systems involves subtleties with respect to both the modeling and the prediction of demand. This is exemplified by the two tweets above,
complaining about opposite rebalancing actions in two different situations: the first complains about not picking up bikes at full stations, while the second complains about the opposite. Depending on the exact nature of demand at a station, it might or might not be optimal to leave a station full rather than remove bikes through rebalancing operations. In systems with hundreds of different stations, the demand patterns of which change over time, distinguishing between these different scenarios (and quantifying the difference) necessarily requires data-driven support systems.

**Estimating Unsatisfied Demand via UDFs**

User dissatisfaction functions (UDFs) underlie almost all of our work with Motivate. These functions provide an inventory model that maps, for each station and any planning horizon, that station’s capacity (i.e., the maximum number of bikes that can be present at that station) and an initial number of bikes at that station at the start of the planning horizon, to the expected number of stock-outs over the course of the horizon. Each UDF is based on a stochastic process from which we sample a sequence of customers arriving to rent or return bikes at that station over the given planning horizon. Then, for every user in the sampled sequence who arrives to rent a bike, the number of bikes (the inventory) decreases by one; for every user who arrives to return a bike increases it by one. However, when the number of bikes is 0 and the next user in the sampled sequence is one who attempts to rent a bike, then we record a dissatisfied user; we similarly record a dissatisfied user when the number of bikes is equal to the station’s capacity and the next user in the sequence attempts to return a bike. In both of these cases, we assume that the dissatisfied customer disappears without a rental/return, so the number of bikes remains at 0 in the first case, and at the station’s capacity in the second case. The UDF is then defined as the expected number of dissatisfied users over the course of the planning horizon. The expectation is
taken over the sampled sequences (for a fixed sequence, the number of dissatisfied users is deterministic). Thus, to specify a UDF, we must also give a stochastic model of user behavior with respect to that station and planning horizon.

We model users wishing to rent bikes through independent non-homogeneous Poisson processes, one for each station. A user’s destination station is independently selected from an origin-dependent distribution, and biking times are independent (their distribution depending only on the origin/destination station pair). With the added assumptions that “upstream” stations never run out of bikes and that no trips begin and end at the same station, the splitting and superposition properties of Poisson processes then imply that the arrival process of users returning bikes to a fixed station is also a non-homogeneous Poisson process that, in addition, is independent of the arrival process of renters at that same station. Moreover, it implies that these processes at different stations are independent. In practice, upstream stations certainly do run out of bikes, but we make this assumption because simulation results (Jian et al. 2016) demonstrate that the resulting prescriptions remain extremely effective even when we relax the assumption and because of the enormous mathematical and computational simplification it yields. In particular, it follows that a city-wide bike-sharing system decomposes by station, i.e., we can compute UDFs for each station in isolation. UDFs can thus be defined for each station and time interval in isolation; they map the initial number of bikes and empty docks at a station to the (expected) number of out-of-stock events over the course of the interval (O’Mahony 2015).

Our choice of objective function, the expected number of out-of-stock events, reflects the subscription-based nature of most large bike-sharing systems, wherein users pay once per year for a subscription and then ride for free throughout the year. Retention of subscribers is then the driving strategic principle behind a successful bike-sharing operation, which
explains why we focus on this particular customer-service focused objective. This objective also aligns with short-term users who pay, e.g., a per-ride fee. In the bike-sharing systems on which we have worked, such day-users contribute a small fraction of rides.

In order to estimate the various parameters of the underlying Poisson processes, transition matrix (of origin-dependent destination distributions), etc., we use a combination of maximum likelihood and specialized decensoring techniques. Some sample UDFs are depicted in Figure 1, where we only show the dependence on the number of bikes, and not the number of docks, for clarity.

**Target Levels for Bikes**

The plots of the UDFs in Figure 1 suggest that the expected number of out-of-stock events is a convex function of the initial number of bikes at each station. It is intuitive that the expected number of renters not finding a bike should be convex (diminishing marginal returns) and decreasing in the number of bikes available at the outset. It is also intuitive that the expected number of riders unable to return a bike due to full docks is also convex and increasing in the number of bikes available at the outset. Sample-path arguments that employ induction on the sequence of events at a fixed station establish that this intuition is,
in fact, correct. There are significant implications for Motivate’s operations. The minimizer of the convex function at each station provides a target level at each point in time, i.e., the number of bikes that minimizes out-of-stock events over a subsequent planning horizon. Motivate uses these target levels in a decision-aid we developed to guide dispatchers over the course of the day (cf. Figure 2). Figure 3 shows how these minimizers vary over time in different neighborhoods; noticeably, stations in the same neighborhood show strong similarities, especially before and at the beginning of each rush hour when the minimizers are clustered either close to empty or close to full, making rebalancing targets conceptually rather simple. In contrast, towards the end of each rush hour, larger differences occur, and rebalancing relies more heavily on analytics.
Figure 3  The optimal number of bikes to position at stations in two NYC regions as a function of the time of day. The East Village, a residential area, wants bikes in the morning, but empty docks to receive returning bikes in the evening. The reverse is true in the Financial District.

System-wide Optimization for Bikes

The UDFs not only provide target-levels for each station, they can also be used to find, at any given time, the optimal allocation of available bikes within the system. We can formulate this optimization problem, for a given time period, as an integer program: the decision variables are the number of bikes at each station, the constraints dictate that the total number of bikes is bounded by a budget, and that the number of bikes at each station cannot exceed the number of docks at the station, and the objective function is the sum of the UDFs over all stations, yielding the overall number of dissatisfied customers. See the appendix for the mathematical formulation.

The convexity of the UDFs allows us to efficiently solve the resulting IP; in particular, the natural linear relaxation is known to have an optimal integer solution. The optimal solutions are visualized for the morning rush hour, for different values of $B$, the total number of bikes in the fleet, in Figure 4. The optimal objective function values for different settings of $B$ quantify the cost impact of changing the fleet size (cf. Figure 5).
Figure 4  The optimal allocation of bikes for the morning rush, for two budgets of bikes. Stations shaded more deeply indicate a larger number of bikes allocated. Left: 5,000 bikes. Right: 10,000 bikes. Map data: ©2018 Google

Figure 5  Objective of the system-wide optimal allocation of bikes evaluated for different fleet sizes $B$; though $B = 11046$ is required to set all stations to their target-levels, the improvement beyond $B = 10,000$ is negligible.
Allocating Capacity

“Cornell came up with a way to very simply measure the impact of a dock’s availability on customers which meant that we could move docks around and before we moved them have a pretty good understanding of the impact on individual customers every single day”, Emily Gates, Director of Operational Strategy, Motivate (2018).

As we incorporated analytics into Motivate’s rebalancing operations, we became more acquainted with demand patterns in Motivate’s systems and found significant potential in the idea of reallocating docks in the system. Specifically, we found that some locations made much better use of their allocated docks than others (cf. Figure 6). To some extent this was to be expected; the dock capacity of most stations in most of Motivate’s systems were set when they were first installed. Thus, the capacity was set before any demand was observed. In this section we describe a methodology, built upon UDFs, that ultimately led to Motivate moving hundreds of docks in its systems.

Integer Programming Model

The integer program to optimize the system-wide allocation of bikes naturally extends to a formulation that allows for reallocated dock capacity; instead of treating the capacities $K_i$ as parameters that are part of the input, we treat them as additional decision variables. To ensure that the IP merely reallocates the dock capacity already present, we require that the total number of docks used by a solution, that is $\sum_i K_i$, be bounded above by the current number of docks. We also introduce a constraint on the total number of docks moved, which is an important practical constraint when modifying a system currently in use; finally, we can also explore the impact of increased dock budgets with the same formulation.

Both technical and practical issues arise with this formulation. From a technical perspective, it is not clear how to optimally solve the resulting nonlinear IP; for example, the
integrality property of the natural relaxation no longer holds. From a practical perspective, it turns out that the optimal allocation suggests moving more docks than stakeholders (Motivate, Department of Transportation, etc.) are willing to approve. Our algorithmic solution addressed both problems simultaneously, as explained next.

**Gradient-descent Search: Local and Global Optima**

The feasible solutions to the IP are given by allocations of bikes and docks across stations that fulfill the budget constraints (for both bikes and docks), as well as bounds on the potential (dock) capacity of each station. One can represent the search space of feasible solutions as an undirected graph in which each feasible solution corresponds to a node in the graph. Two nodes corresponding to two feasible solutions are adjacent if one can be obtained from the other by reallocation of at most one dock and at most one bike. With this definition of neighboring feasible solutions, one has a corresponding notion of a “local optimum” – a feasible solution with objective function as good or better than any of its neighbors. Our optimization algorithm is an intuitive discrete analogue to gradient-descent
algorithms commonly used in continuous optimization: starting at any given feasible solution, the algorithm repeatedly updates the current solution to be the best solution within the neighborhood of the current solution. We can prove that if we start at a node corresponding to the optimal bike-allocation for the current dock-allocation, and repeatedly update until we reach a local optimum, then after $k$ such updates we obtain the best possible solution that can be obtained by moving at most $k$ docks. This allows us to solve the integer program with the additional constraint bounding the number of docks moved.

The proof of this result relies on a more general property of the user dissatisfaction functions, called multimodularity, which was introduced by Hajek (1985); although the precise definition is somewhat involved, one can view this property as a kind of multi-dimensional diminishing-returns property. For example, if we consider a station for which the current allocation is 10 empty docks and 10 full docks, and consider the improvement gained by adding one full dock (that is, a bike and a dock), then that improvement is at least as much as is gained by adding a full dock to the same station already allocated 11 empty docks and 10 full docks. An inductive proof over all sample paths, along the same lines as was used to establish convexity just in the number of bikes, can be used to show that the UDFs are multimodular.

The intuition underlying the result that in the $k$th iteration we find the best allocation obtainable by moving at most $k$ docks is based on a weaker statement, that is, that without the constraint on the number of docks moved, the local optimum found is a global optimum. The proof of this statement is instructive, and illustrative of the way in which we exploit multimodularity. Suppose, for a contradiction, that a local optimum is not a global optimum. Then find another feasible solution in the graph with a better objective function value, and among all such solutions, choose the feasible solution closest to the
local optimum in the graph (where “closest” is defined with respect to the number of edges in the shortest path between them); if there are multiple such nodes that are equally close, choose one arbitrarily. Suppose that \( u \) is the node corresponding to the local-but-not-global solution, and \( v \) is that closest node just selected. The node \( v \) need not be a global optimum either; it is simply a node that is better than \( u \). Consider the shortest path between nodes \( u \) and \( v \), and let \( w \) be the node adjacent to \( v \) that is traversed in this path just prior to reaching \( v \). By the choice of \( v \), the objective function value of \( w \) is worse than it is for \( v \); in other words, the local move made in changing the solution at \( w \) to become \( v \) is an improving one. However, the multimodularity property allows us to argue that making the same change to node \( u \) must yield at least as much improvement, contradicting the local optimality of the feasible solution corresponding to node \( u \). The same ideas can then be extended to guarantee that the solution we find in the \( k \) iterations is globally optimal when restricting ourselves to moving at most \( k \) docks, even though in that case it is not in general true that local optimality guarantees global optimality.

**Best in \( k \) Iterations**

The discrete gradient-descent algorithm we derived is significantly more efficient than relying on general-purpose integer optimization techniques. Moreover, we were able to provide Motivate with a complete package to run analyses from parameter estimation to visualizations of optimal solutions, partly because we do not rely on general-purpose IP solvers. But most importantly, the fact that \( k \) iterations of the algorithm yield *the best allocation that can be obtained by moving at most \( k \) docks* addresses practical concerns about moving more docks than stakeholders will countenance. By running the algorithm for only \( k \) iterations, we obtain the optimal solution with the additional constraint on movement of docks; this broadens the scope of results obtainable by the tools of discrete
convexity. As a consequence, we find that even though the optimal allocation may require moving thousands of docks (in NYC), significant impact can be had by moving only a few hundred docks.

**Algorithmic Efficiency**

A careful implementation of the algorithm allows us to rapidly solve the IP on real instances, yielding optimal solutions within minutes. While one element of this improved implementation involves only a straightforward use of appropriate data structures, a more fundamental change is based on generalizing the structure of the gradient-descent procedure. The gradient-descent algorithm makes a unit bike/dock change to the system in each iteration; instead, we can consider a series of phases in which the algorithm starts by making “big” gradient-descent steps (by changing bike/dock allocations in units of $2^\ell$ bikes/docks instead of one at a time), repeatedly finding a local optimum with respect to the coarser step and using that result as an initial solution with respect to the next level of refinement (say, changing by $2^{\ell-1}$). These ideas lead to both empirical and (non-trivial) theoretical improvements to the running time achieved by our approach.

**Robustness**

Docks are not as easy to move as bikes; they are heavy pieces of equipment that require a crane to lift them onto a flatbed truck. Consequently, in making a recommendation to move a substantial number of docks, we wanted to be sure that these recommendations were consistently supported by a number of different analyses. In particular, one should think of dock changes as being made on an annual basis, and therefore should be robust to any seasonality effects in the demand patterns. Furthermore, these changes are made in the context of ongoing expansion to the footprint of the system, which could also affect the demand pattern at previously existing stations. For example, in NYC, a series of system
expansions have increased the number of stations in the city from about 330 stations in 2015 to more than 700 stations in 2018 over a substantially increased geographic footprint. Such great changes to the system could easily change the demand structure in ways that would contradict the recommended changes.

We have used the discrete gradient-descent algorithm to evaluate the impact of proposed changes on demand data stemming from many different months, including from different years and different seasons. Significantly, the improvement in the objective obtained by the proposed changes was robust with respect to these different demand estimates. Even in NYC, with its frequently changing footprint, we found that optimization-determined reallocations based on data from one year continue to yield great improvement under demand estimates from different years. For example, in Figure 7, we display the improvements for spring and fall 2017 that are obtained by reallocating based on data from summer 2015.

Beyond considering robustness with respect to demand patterns, Motivate expressed concern about an issue not captured by the optimization problem described above: effectively, the IP optimizes the service quality for a system with perfect bike allocations. Bike allocations are primarily adjusted through overnight rebalancing, so the optimization model effectively assumes perfect overnight rebalancing. In practice, rebalancing is limited, and a goal of this project was to reduce the need for rebalancing. To account for this shortcoming, we defined an alternate objective: a long-run average of the user dissatisfaction function that computes the expected number of out-of-stock events not over the course of one day, but on average over the course of infinitely many days. In many ways, this is the diametrically opposite philosophy to the one we have adopted thus far – it models the expected number of out-of-stock events at a station with no rebalancing at all. By optimizing with the different objectives, we obtain two different “optimal” solutions. One can
construct examples of distributions over demand for individual stations such that when evaluating one optimal solution with the other objective, the performance is arbitrarily worse than the optimal solution for the other. Surprisingly, we do not find this to occur on real data: optimizing for one objective gives a near-optimal solution for the other. It is worth emphasizing that this is a property of the data, not of the model. Even more surprisingly, this is not due to small $L_1$ distance between the two “optimal” solutions or due to the objective being particularly flat near the optimum. Instead, it is a function of both objectives being flat in the direction of the other optimal solution. In particular, the optimal solutions (in the two regimes) perform, in either regime, significantly better than the current allocation (cf. Figure 7), despite being about two thirds as far away from each other (in $L_1$ distance) as they are from the current allocation.
Implementation and Evaluation

After a long planning period, a pilot project in NYC entailing the movement of 34 docks went ahead in November 2017. The pilot allowed us to run a counterfactual analysis based on realized demand: for each of 3 stations that had docks added, we computed for every weekday in April 2018 (given the sequence of arrivals that occurred over the course of the day) the number of out-of-stock events that would have occurred with the same demand if the docks had not been added (and no additional rebalancing had been performed). This can be achieved in a purely data-driven manner, without the need to resort to stochastic-process modeling and the attendant data-fitting issues, as we proved. We also evaluated the increase in the number of out-of-stock events at the 3 stations that had their capacity reduced, though in that case, one must rely on fitted stochastic models.

The resulting analysis (cf. Figure 8) indicates that the reduction in out-of-stock events per dock added to stations averaged about 1.5 per day, while the increase in out-of-stock events at stations with reduced capacity averaged about 0.08 per dock per day. As such, each dock that was moved reduced the need for rebalancing (with average service quality kept constant) by more than 1.42 bikes per weekday. Translating this improvement into reduced rebalancing needs, annual savings of tens of thousands of dollars were realized, even though the pilot moved only a tiny fraction of all docks. In systems across the United States, Motivate has deployed the same optimization approach to guide the reallocation of hundreds of docks, whilst continuously monitoring the resulting impact on out-of-stock events. Comparing that impact to that of rebalancing, and taking into account the cost of both rebalancing of bikes and reallocations of docks, the cost of moving docks generally pays off in as little as two to five weeks.
Bike Angels

One natural approach to modulate demand for a bike-sharing system would be to adopt a framework of dynamic pricing and by doing so, provide a means to circumvent the need for rebalancing. This approach has been adopted (with both great effect and public derision) in the related realm of ride-sharing. For Motivate, such an option is not feasible, since all of its systems are obliged (by its city partners) to offer customers annual subscription plans; these plans effectively offer customers *all you can ride* for a low price. For example, in NYC an annual plan costs only a little more than a one-month pass for the Metropolitan Transportation Authority. In 2015 we suggested to Motivate that despite the annual plans, systems could modulate demand by providing incentives to encourage rides that are beneficial for system balance.

The proposed incentive program was based on a map that labels each station as being in one of three different classes: *neutral*, *return*, or *rent*. Given that map, customers were to receive points for trips undertaken from *rent* stations to either *neutral* or *return* stations and for trips from *neutral* stations to *return* stations (where trips from *rent*
to return stations receive double points). As such, the set-up involved two major design decisions: how do we label each station at a given point in time, and what rewards should customers receive for the collected points — we focus here on the former.

In the first trial of an incentive program, we had fixed labels for each station in each rush period, that is, we decided up front on the neutral/return/rent label for each station and kept those labels fixed throughout the rush period, and the same labeling was used for each weekday. In terms of user experience, there is an obvious advantage to such static labels: customers can plan on the same trip getting them the same number of points every day. In terms of efficiency, however, setting labels up front comes at a cost: given the randomness in daily usage, the program sometimes awarded points for trips that failed to improve the system balance.

In order to quantify potential failures in efficiency, we need a metric to evaluate the impact of each rental and each return. For ease of exposition, we restrict attention to returns. To derive such a metric we compute, for each incentivized return, the change it caused in the UDF for the return station, that is, the difference between the value of the UDF before and after the return. In essence, we compute the discrete derivative of the UDF at the station with respect to one additional bike (cf. Figure 9). One can show through a sample-path argument that this difference is always bounded between -1 (worst possible: the return increased the number of future out-of-stock events by one) and 1 (best possible: the return decreased the number of future out-of-stock events by one). Figure 10 visualizes the computed changes for both rentals and returns incentivized by the static program. We find there that (i) most incentives, by far, were given for rentals/returns that have positive impact and (ii) a nontrivial portion of incentives went to rentals/returns that have no positive or even a strictly negative impact.
Figure 9  Discrete derivatives for each of the four UDFs in Figure 1.

Since the discrete derivatives can be computed in real time, we could also use them to guide decisions on where/when to incentivize; in particular, this would guarantee that...
incentives are given if and only if the rental/return reduces the expected number of future out-of-stock events. However, setting the labels in real time comes at a cost in terms of user experience: because the status of a ride is not determined even at the time of the rental, the user must make her decisions without knowledge of whether any particular return will be rewarded. As such, it is natural to investigate the tradeoff between efficiency on the one hand and predictability of incentives on the other.

In our study (Chung et al. 2018), we used the data-set from the static program to investigate how frequently relabeling needs to occur to ensure that near-perfect efficiency is maintained. With that data, relabeling stations every 15 minutes suffices to ensure that exactly the right rentals/returns are incentivized. As one further increases the length of the
time intervals, the efficiency degrades smoothly. For example, relabeling every 60 minutes retains more than 97.5% efficiency (cf. Figure 11). This led Motivate to adopt an incentive scheme in which the discrete derivatives of the UDFs, updated on a quarter-hourly basis, dictate whether or not a station is incentivized.

Our work with Motivate originally proposed incentives, set up the analytics for the first pilots, identified inefficiencies with the static scheme, and informed the design of today’s program. The impact of the Bike Angels program on service quality in NYC now matches that of motorized rebalancing, but at a much lower financial and environmental cost; furthermore, Motivate has also launched Bike Angels in its Ford GoBike program in the San Francisco Bay Area and in its Capital Bikeshare program in the Washington metropolitan area.

Conclusion

“So much of our decision-making was happening with intuition and the work with Cornell frankly was the foundation of this pivot for this company from an operating company working with intuition to an operating company working with data.”

Jay Walder, CEO, Motivate (2018)

Our work with Motivate introduced UDFs throughout their enterprise. UDFs now inform the system design with respect to station sizes, they guide dispatchers in deciding how to route vans and box trucks to rebalance bikes, and they power the Bike Angel incentive schemes in NYC and elsewhere.

But our work with Motivate extends well beyond these projects highlighted in this paper, since there are a multitude of opportunities to optimize operations. For example, a subtle extension of the sample-path arguments used to establish UDFs for single-station performance is at the core of a model that determines an optimal allocation of “trikes”
that have been used for non-motorized mid-rush-hour rebalancing (cf. Figure 12); as a consequence, the problem reduces to finding a minimum-cost matching of cardinality $k$ in bipartite graphs (which can be easily solved). Furthermore, integer programming models were used to run pilot studies for optimized overnight truck routing for rebalancing given time-limited capacity to move bikes with a handful of trucks. Finally, optimization models were employed in the allocation of valets to staff corrals, where targeted stations had, in effect, expanded surge capacity during the day (cf. Figure 13).

Having a single measure across Motivate’s operations in NYC allows the company to quantify the relative impact on service through apple-to-apple comparisons. Further, many of these methods have been exported from NYC to their other systems nationwide, putting Motivate’s operations across the country on a sophisticated analytical footing.

Bike Angels and the dock reallocation efforts use UDFs in particularly creative ways to not only improve customer access to the system, but also to save costs for Motivate. Further, they do so in an environmentally sustainable way. Conservative estimates on these costs show that using vehicular balancing to achieve the same effect of even just Bike Angels in NYC would cost over $1,000,000 per year and create an additional 500 tons of CO$_2$ emissions per year. The extent to which Bike Angels has become part of popular culture in NYC is reflected in the short documentary, “The Point of a Ride” (Gerard 2018), which premiered at the 2018 Bicycle Film Festival in New York, and news stories (NPR 2018) about individual successful Bike Angels participants. Finally, they gave rise to intriguing mathematical challenges, which led to the development of new algorithmic results with potential applications beyond bike-sharing.

**Acknowledgments**

This work was supported in part by NSF grants CCF-1740822, CCF-1526067, CMMI-1537394, CCF-1522054, CMMI-1200315, CCF-0832782, and CCF-1017688 as well as Army Research Office grant W911NF-17-1-0094.
Figure 12  A trike used by Citi Bike in New York City to transport up to 5 bikes.

Figure 13  Part of a staffed corral in New York City; by using all physical space available, the capacity is extended far beyond the number of docks.

It would not have been possible without Motivate’s commitment to becoming a data-driven company and their openness to trying new things.

Appendix. Optimization Formulations

Denote the number of docks at Station $i$ by $K_i$, the number of bikes by (a decision variable) $b_i$, the UDF by $c_i(b_i, K_i)$, and the total number of bikes available by (an input parameter) $B$, the nonlinear integer program
discussed in the paper is

$$\begin{align*}
\text{minimize}_x & \sum_i c_i(b_i, K_i) \\
\text{s.t.} & \sum_i b_i \leq B, \\
\forall i : & 0 \leq b_i \leq K_i.
\end{align*}$$

When we discuss the system-wide optimization for bikes, the value $K_i$ giving the number of docks at Station $i$ is viewed as a fixed input parameter. Later, when dock moves are considered, $K_i$ becomes a decision variable in addition to $b_i$, and the formulation is augmented with a constraint on the number of available docks. In that case, we can denote the current number of docks at each station $i$ as $\bar{K}_i$, which allows us to also write the constraint

$$\sum_i |K_i - \bar{K}_i| \leq 2k$$

to ensure that at most $k$ docks are moved.

References


Freund, Daniel, Shane G. Henderson, David B. Shmoys. 2017. Minimizing multimodular functions and allocating capacity in bike-sharing systems.


