Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Data Analysis and Optimization for (Citi)Bike Sharing

Eoin O'Mahony[1], **David B. Shmoys**[1,2]
Cornell University
Department of Computer Science[1]
School of Operations Research and Information Engineering[2]

Abstract

Bike-sharing systems are becoming increasingly prevalent in urban environments. They provide a low-cost, environmentally-friendly transportation alternative for cities. The management of these systems gives rise to many optimization problems. Chief among these problems is the issue of bicycle rebalancing. Users imbalance the system by creating demand in an asymmetric pattern. This necessitates action to put the system back in balance with the requisite levels of bicycles at each station to facilitate future use. In this paper, we tackle the problem of maintaing system balance during peak rush-hour usage as well as rebalancing overnight to prepare the system for rush-hour usage. We provide novel problem formulations that have been motivated by both a close collaboration with the New York City bike share (Citibike) and a careful analysis of system usage data. We analyze system data to discover the best placement of bikes to facilitate usage. We solve routing problems for overnight shifts as well as clustering problems for handling mid rush-hour usage. The tools developed from this research are currently in daily use at NYC Bike Share LLC, operators of Citibike.

1 Introduction

Bike-sharing systems are a cost effective way of promoting a sustainable lifestyle in urban areas. The number of bike-sharing systems has more than doubled since 2008 (Larsen 2013). These systems generally consist of stations where users can take out or drop off bikes, and may return a bike to a free dock at any station. New York City launched the largest bike-sharing system in North America, Citibike, in May 2013 with over 300 stations and 5000 bikes. The system has been a success with ridership approaching 40000 trips per day. With this success comes a set of management problems. Chief among these is the issue of system imbalance; bikes become clustered in certain geographic areas which leaves other areas devoid of bikes; for example, the traders wake up early in their East Village apartments, rapidly deplete the supply of bikes there, and then overwhelm the capacity for docks in the Wall Street area. This system imbalance necessitates moving bikes around the city to put the system back in balance. This is achieved either by trucks, as is the case in most bike-share cities, or other bicycles with trailers, as is being tested in New York.

Operators of bike-sharing systems have limited resources available to them, which constrains the extent to which rebalancing can occur. Hence, this domain is an exciting application for the field of computational sustainability. Based on a close collaboration with NYC Bike Share LLC, the operators of Citibike, we have formulated several optimization problems whose solutions are used to more effectively maintain the pool of bikes in NYC. There is an expanding literature on operations management issues related to bike-sharing systems, but the problems addressed here are particularly suited to the complex blend of human and system constraints that are present for Citibike. For these problems, we shall present results utilizing different approaches: integer programming formulations that can typically be solved (at scale) by off-the-shelf integer programming solvers, and heuristic approaches that yield good solutions quickly.

We begin by tackling the problem of how best to use system data to plan for usage. That is, we want to place bikes at stations to handle the surge in usage experienced during rush-hours. We approach the issue of inferring true demand for trips and use these amounts to better plan for system usage. From these computations we know both when and where bikes are needed and progress to answering the question: how do we get them there?

We focus on two very different rebalancing problems. First, we tackle rebalancing the system during rush-hour, developing novel methods for optimizing rebalancing resources. During rush-hour, system usage is high, rendering large truck routes unreliable as the system state might shift dramatically before the route is completed. Traffic is also at its peak during rush hour, which greatly limits the ability of trucks to move easily through the city; this motivates the use of bike trailers instead of large trucks. The nature of the mid-rush balancing requires a drastically different approach than the one used for the overnight problem. Our goal is not to maintain system balance but to ensure that users are never too far from either a bike or a dock. To achieve this, we formulate a clustering problem that yields stations in the city for which rebalancing resources can be targeted.

We then tackle the problem of moving bikes around the city overnight. Overnight the system experiences low usage

and as a consequence stockage levels are relatively constant. Traffic is also at its lowest during these hours resulting in trip times being both reliable and short. This allows a rebalancing plan spanning the overnight shift to be computed and executed without fear of users making it redundant or counterproductive. We formulate an optimization problem whose goal is to produce a series of truck routes to get the system as balanced as possible during the overnight shift. We provide both theoretical results as well as an empirical approach to this problem based on a (relatively) tractable integer programming formulation. The combination of these two approaches yields a fast method of obtaining high quality solutions that can be used in practice.

2 Related Work

Due to the increasing importance of bike-sharing programs, and the operational difficulties in managing them, a great deal of attention has been focused on a variety of problems that relate to bike-sharing. In particular, work such as (Raviv, Tzur, and Forma 2013), (Contardo, Morency, and L.-M. Rousseau 2012), (Schuijbroek, Hampshire, and van Hoeve 2013), (Chemla, Meunier, and Calvo 2013) and (Shu et al. 2013) focus on the problem of (overnight) rebalancing. (Raviv, Tzur, and Forma 2013) tackles the problem of finding truck routes and plans for how many bikes to move between stations. The paper minimizes an objective function tied to both the operating cost of the vehicles as well as penalty functions relating to station imbalance. The models provided are benchmarked on instances from both the Paris and Washington DC bike-share systems. (Schuijbroek, Hampshire, and van Hoeve 2013) combines both finding service level requirements for stations with planning truck routes to keep stations rebalanced. They use a clustering based heuristic for routing on data from Boston and Washington DC to produce truck routes. (Rainer-Harbach et al. 2013) take a local search approach to finding both routes for trucks and the number of bikes to be collected or dropped off at each station. (Contardo, Morency, and L.-M. Rousseau 2012) identify that a different rebalancing approach needs to be taken during rush hours. They formulate flow problems on space-time networks. Solutions are generated using a combination of Dantzig-Wolfe decomposition and quickly generated upper and lower bounds. (Chemla, Meunier, and Calvo 2013) solves the static rebalancing problem, where a plan of where to move bikes is created. They provide a branch-and-cut algorithm for a problem realization and a tabu search to find heuristic solutions. (Shu et al. 2013) uses a time-flow formulation combined with stochastic modeling to tackle the problem of rebalancing. These papers tackle the rebalancing problem in a way that is similar to traditional inventory management and package routing problems, for example (Anily and Bramel 1999), (Archetti et al. 2007) and (**?**). However, in our work we approach the problem in a manner closer to orienteering problems (Vansteenwegen, Souffriau, and Oudheusden 2011), (Chekuri, Korula, and Pál 2012). Our work also handles full size instances; both in terms of number of stations and trucks considered. Much of the previous work has focused on instances for which two or three trucks are available. Furthermore, our work builds on the understanding of the practicalities of running Citibike, in their current state, it is much simpler operationally to simply have trucks go to an overloaded station, fill the truck with bikes, and then deposit all of them at a (sufficiently) depleted station.

Other work creates models to analyze bike-sharing systems, specifically (Nair et al. 2013), (Vogel and Mattfeld 2011), (Kaltenbrunner et al. 2010) and (Lin and Yang 2011). This work focuses on modeling how users will impact the system, detecting usage patterns from behavior. This insight into usage patterns is used to create stochastic models representing system usage. (Vogel and Mattfeld 2011) classifies stations based on their usage patterns, identifying stations used by commuters, tourists, etc. (Garca-Palomares, Gutirrez, and Latorre 2012), (Martinez et al. 2012) and (Romero et al. 2012) aim to optimize the placement of stations in bike-sharing systems.

Our work provides fundamentally different models of bike rebalancing compared to previous approaches: for mid rush-hour rebalancing, we focus on a covering problem viewpoint, closely tied to the very small number of pairs of stations that can be rebalanced by bike trailers; for overnight rebalancing, we focus on full truck-load routes that give rise to an problem of covering a bipartite graph with sufficiently short alternating paths. These models are driven by observations obtained by our collaboration with New York Bike Share LLC., the operators of Citibike.

3 Data Analytics for Bike Share Systems

Our earliest goal for the collaboration with NYC Bike Share LLC was to make their planning and decision making *data driven*. Specifically, it is crucial to use the data available to understand how the system is being used and where usage is putting stress on the system. The first problem we tackled was the problem of rush-hour planning. Weekday rush-hours (6am-10am and 4pm-8pm) account for the majority of bike trips taken in New York. Although mostly symmetric over the course of a day, each rush-hour period in isolation is highly asymmetric. In fact we observe many extremes of behavior with some areas of the city having a large outflow of bikes and other areas having a large inflow of bikes. To deal with this surge we aimed to answer the following question: suppose we could click our fingers and have the system in any state before the morning and evening rush hours, what would that state be? To answer this question we used a number of methods, each of which raises deeper questions about the analysis of bike-share usage.

Planning for Rush-Hour Usage To plan for a rush-hour we need to know where we expect bikes to be taken from and areas where we expect they will accumulate. We also need to identify stations that are *self-balancing*, specifically their flow of bikes *in* is roughly equal to their flow of bikes *out* thus requiring no rebalancing actions. Out first approach to discover the ideal system state before the morning and evening rush-hours relied on clustering stations based on their observed usage. The intuition is that stations that experience similar behavior during rush-hours will belong to the same cluster, we then analyzed the type of behavior typ-

ical of each cluster and label the cluster with a desired level. These levels are used to inform the operators of Citibike where bikes need to be placed to anticipate user demand.

This approach has been highly successful, by using the fill levels generated by this computation the operations team at NYC Bike Share have been able to tailor their rebalancing operations to best serve the heavy rush-hour usage. However, although successful this method has a number of limitations. Specifically there is a weakness related to exploitation versus exploration. This method is slow to react to shifts in usage, to illustrate this we will consider an example that happened at a station in Brooklyn. The station was initially self-balancing with low usage, thus it was assigned a low fill level. However, over time the behavior of this station shifted to become a consumer of bikes. This change was not reflected in its level as there were never enough bikes placed there for high usage to be recorded. This station was identified by chance and this experience led us to believe that there might be others in a similar situation. This caused us to consider the question, assuming we could keep each station stocked with bikes and spaces, what would a typical day's usage look like? This question is, in essence, what is the true demand for bikes in the system?

Computing Demand for Bikes Although system data gives us information on each ride taken by users in New York these data might not be a true reflection of the actual demand in the system. Consider a bike-share station at Penn Station in the evening rush hour. A huge number of commuters want to return bikes and take a train from the station. However if we were to not rebalance this station it would quickly fill up and we may observe a fraction of the trips that could have happened. This motivates the computation of the underlying demand. Knowing the demand for bikes and docks in the system allows us to plan more effectively for usage.

Observed trip data differs from the true demand due to censoring, that is stations being empty or full preventing users taking or returning bikes at the station. For many days we may observe zero trips for a time period but perhaps this is related to the station being empty/full. These outage windows are highly consistent as the morning and evening rush-hour behavior patterns are similar from day to day, meaning that the same stations are empty at the same time almost every day. However due to rebalancing operations we have days where we managed to replenish these stations with bikes or remove excess bikes. Out aim is to rely on rebalancing operations having had sufficient impact to give us data for most stations.

Consider a matrix of observations, O,where O_{ij} is equal to the number of bikes out on day i at time j and a level matrix, L, where L_{ij} is the number of bikes at the station on day i at time j. We use the average number of trips for each time window as a lower bound on the true demand. The lower bounds we produce are first order approximations of the true demand. Consider the impact of changing operations to facilitate these demands, this could easily drive more ridership as people see the system becoming more reliable.

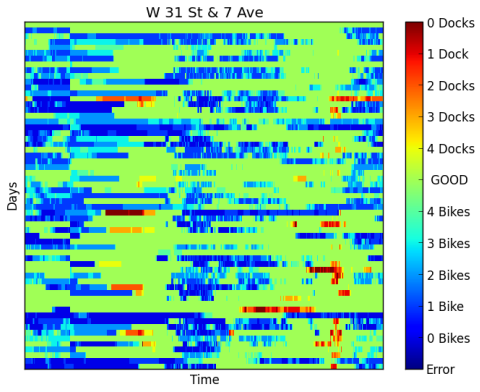

Figure 1: Example mask matrix for a Penn Station bike-share station over 60 days.

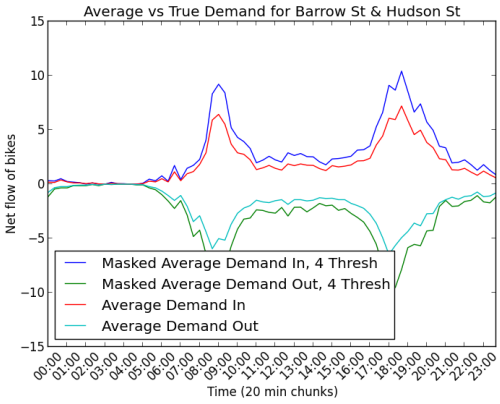

Figure 2: Example of the two lower bounds on true demand for a station in the West Village.

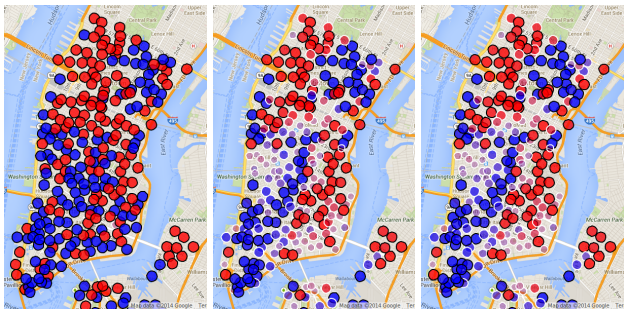

Figure 3: Morning levels assigned based on different cost functions. From left to right, minimizing max gap, minimizing the sum of the gaps, minimizing the sum of the gaps squared all with six thousand bikes. The color of stations corresponds to fill percentage, blue to 10%, red to 90% and purple to 50%.

To compute more accurate lower bounds on demand we need to take censoring of demand into account; to do this we mask the observed trip matrices, removing elements where the corresponding level element is at zero. That is lower bound on true demand of bikes out at time j, $T_j^o = avg_i(O_{ij}|L_{ij} > 0)$ This in essence ensures that zeroes that occur in the observed matrix are actual zeroes and not just zeroes due to outages. Due to broken bikes and broken docking points we use a soft outage number, that is considering a station to have a stockout if the number of bikes or docks drops below a given threshold. In practice, the correct value for this threshold is between two and five. An example of such a mask matrix for one of the Penn Station locations is shown in Figure 1. Using a threshold of four, the bounds on true demand flows for a West Village station are shown in Figure 2. From this figure we can see that the lower bound obtained by the mask matrix is larger in both magnitude and duration throughout the rush-hours.

Optimizing Resource Allocation Under the assumption that the lower bounds we have computed are a good proxy for the underlying user demand, optimization of management decisions for the system becomes possible. We can frame the question of how many bikes to place at a station before the rush hour as an optimization problem. Given, for each station, the expected number of bikes in and out of the station for each minute of the rush hour, the station capacities and the number of bikes that can be deployed into the system we solve an optimization problem where for each station we decide the number of bikes placed there, $X_s \in \{0...c(s)\}$ (where $c(s)$ is the capacity of the station). Given a budget of bikes B we need to ensure that $\sum_s X_s \leq B$. We then minimize some objective function $J(X)$, the structure of which impacts the quality of the solution to the problem. One initial candidate is to compute the net flow from the lower bounds for each station and look at the smallest and largest value for this curve over the rush hour. These values are the maximum imbalance the flow at the station will create. We can then penalize a station's level for being too far under or over these values. This penalty can take the form of minimizing a quadratic function of the differences, minimizing the sum of the differences or minimizing the maximum difference. Given for each station, s, the net flow curve at each time t as f_t^s, minimizing the maximum difference can be solved by the integer program below.

$$\min D \tag{1}$$

$$s.t. \quad \sum_s X_s \leq B \tag{2}$$

$$\forall s \quad D \geq X_s + \min_t f_t^s \tag{3}$$

$$\forall s \quad D \geq c(s) - X_s - \max_t f_t^s \tag{4}$$

$$\forall s \quad X_s \in \{0...c(s)\} \tag{5}$$

$$D \geq 0 \tag{6}$$

Minimizing the sum of the differences and the sum of the differences squared take similar forms with auxiliary variables recording the amounts that each station is under- and over-served, i.e., by constraints (3, 4). These auxiliary variables are then summed or squared and summed. It is important to put a lower bound of zero on the difference variables since we concern ourselves only with trips missed. Solutions to the optimization problem produced by Gurobi (Gurobi Optimization 2014) using these three objective functions are shown in Figure 3. Both minimizing the sum and sum of squared worse case imbalances result in a solution that matches operator intuition about the system.

4 Rebalancing

Having computed the desired fill levels for stations before the morning and evening rush-hours, as well as analyzing their behavior during these rush-hours, we address the problem how to get them there? We take different approaches for both planning for the rush-hour surge and managing this surge in ridership. Both our approaches are novel and drastically different from existing work.

Mid-Rush Rebalancing

The morning and evening weekday rush-hours account for the majority of trips taken on New York's bike-share system. Usage is extremely high during these periods and asymmetric; that is, the net flow of bikes out of many stations is largely positive or largely negative and often matched by an opposite symbol flow in the complementary rush-hour period. From observation of data, the net flow of stations can be computed. The behavior of stations from one day's rush-hour to the next is very consistent, allowing us to classify them as either producers, consumers or balanced stations.

Our goal during the rush hour is to ensure that users of the system are not too far from either a bike or a dock. A criterion close to this is contained within the contract that requires the operator of Citibike to maintain a specified level of quality of service in a range of aspects; fines are imposed for failing to meet certain levels. We will focus rebalancing resources on covering the critical areas of the city and will be able to rebalance only a small subset of stations. Using historical data that indicate which stations accumulate bikes (producers) and which lose bikes (consumers), we want to ensure that each producer station is close to a producer station that will be rebalanced, and that each consumer station is close to some rebalanced consumer station. To rebalance a consumer station, bikes must be delivered to it, ideally from a producer station where bikes need to be removed for rebalancing. We select producer and consumer stations to rebalance, pairing them up so that rebalancing is achieved for both producer and consumer simultaneously.

In NYC, most of the mid-rush rebalancing is done by special bicycles outfitted with trailers that can hold a few Citibikes, typically three; due to the nature of this resource, the most effective plan is to designate certain pairs of stations (i.e., with one producer and one consumer) to be targeted for mutual rebalancing. These pairs must be sufficiently close, so that the bicycles can be moved effectively within the narrow timeframe of a quickly transpiring rush hour. These considerations gave rise to the following optimization model:

Definition *Mid-Rush Pairing Problem* We are given a complete undirected graph $G = (P \cup C, E)$ with a non-negative metric distance function $d(e), \forall e \in E$, as well as an integer k, and an integer T; the goal is to select

two subsets, $P' \subseteq P$ and $C' \subseteq C$ such that $|P'| = |C'| \leq k$, such that there exists a perfect matching in $\{e \in E : d(e) \leq T\}$ between P' and C' which minimizes $\max\left(\max_{p\in P} d(p, P'), \max_{c\in C} d(c, C')\right)$ We define $d(p, P')$ to be the distance between p and the closest point in P' to it.

Integer Programming Formulation Similar to the case of the k-center problem, (Hochbaum and Shmoys 1985), the objective function is determined by exactly one edge length. Thus there are only a polynomial number of possible optimal values - $|P|^2 + |C|^2$. For each potential optimal value, we either verify its infeasibility (by showing that the IP is infeasible) or we have a solution.

$$\sum_{u\in V} x_u \leq k, \qquad \forall V \in \{P, C\}; \tag{7}$$

$$\sum_{u\in V:(u,v)\in E_V} y_{(u,v)} = 1, \qquad \forall v \in V, \quad \forall V \in \{P, C\}; \tag{8}$$

$$y_{(u,v)} \leq x_u, \qquad \forall (u, v) \in E_V, \quad \forall V \in \{P, C\}; \tag{9}$$

$$\sum_{c\in C:(p,c)\in E_M} m_{(p,c)} = x_p, \qquad \forall p \in P; \tag{10}$$

$$\sum_{p\in P:(p,c)\in E_M} m_{(p,c)} = x_c, \qquad \forall c \in C; \tag{11}$$

$$x_u \in [0, 1], \qquad \forall u \in V; \tag{12}$$

$$y_{(u,v)} \in \{0, 1\}, \quad \forall (u, v) \in E_P \cup E_C; \tag{13}$$

$$m_{(p,c)} \in \{0, 1\} \qquad \forall (p, c) \in E_M. \tag{14}$$

In the integer linear programming formulation of the Mid-Rush Pairing Decision Problem, we wish to decide if there is a feasible solution in which each producer is served within a distance of d_P and each consumer within d_C, and that the paired nodes are within input threshold T; we let E_P be those pairs $(u, v) \in P \times P$ for which $d(u, v) \leq d_P$, E_C be those pairs $(u, v) \in C \times C$ for which $d(u, v) \leq d_C$, and let E_M be those pairs (u, v) such that $d(u, v) \leq T$. In the IP formulation, there exists a 0-1 decision variable x_u for each $u \in P \cup C$ to indicate whether (or not) that node is selected as one of the $2k$ paired stations; there is a 0-1 decision variable $y_{(u,v)}$ for each $(u, v) \in E_P \cup E_C$ to indicate whether node v is served by node u in the pairing. Finally, we have a 0-1 decision variable $m_{(p,c)}$ for each $(p, c) \in E_M$ to indicate whether producer p and consumer c are paired. Hence, we get the integer programming relaxation as shown above. We implemented the above model in Gurobi, by using a bisection search over the space of possible objective functions we were able to solve instances from New York in under a minute. We believe this performance to be due to the high quality of the LP relaxation of the above IP, in most cases the LP relaxation is integral.

Overnight Rebalancing

The majority of rebalancing operations occur overnight. From our analysis of system data and underlying demand we have computed the desired state of the system for start of the morning rush-hour. Previous work has attempted to compute routes for rebalancing trucks that allow a fully general pattern of pickups and drop-offs, and specify the number of bikes to be collected from and dropped to each station. We take a different approach that is motivated by working closely with the operators of the New York bike-sharing system. We restrict ourselves to moving truckloads of bikes. With this restriction, we formulate a model to optimize the use of a given-size fleet of rebalancing trucks. We derive an IP formulation that is reasonably tractable for fleets with a small number of trucks, and then provide a heuristic approach that takes advantage of the fact that the IP finds high quality solutions for the 1-truck special case.

During the overnight rebalancing shift, the goal is to get the system ready for the morning rush hour. We aim to have all stations at their desired level as specified by a balancing plan. Often this is an unrealistic demand as the resources available are inadequate to achieve this. This motivates the problem of getting the system as close as possible to this state with the resources available. To achieve this, we compute a set of routes for rebalancing trucks that optimize the number of stations rebalanced. We limit these routes to move only full truckloads of bikes. Previous approaches have focused on the number of bikes to move between stations. From analyzing system state at the beginning of the overnight shift we observed that it is desirable to move only full trucks of bikes. A full truck of bikes is, in most cases, enough to bring a station to the required level. Also, the travel time in Manhattan can dominate the loading time of a truck, making it desirable from an operational standpoint to move only full truck loads of bikes. Finally, the simplicity of the instructions needed to implement full truckload routes is an important element in the practicality of this approach. Using these observations, we formulate the problem as trying to find a set of truck routes that rebalances as much of the system as possible in the time available. We route trucks in a bipartite graph, where one node set consists of stations with a surplus of bikes and the other of stations with a deficit of bikes. We now formally define the *Overnight Rebalancing Problem*. The intuition for this model is that we want to have routes for trucks that alternate between surplus and deficit stations. The distance between a surplus and a deficit station takes into account both the travel time and loading/unloading time. The time limit is determined by the length of a shift operated by rebalancing staff.

Definition *Overnight Rebalancing Problem* We are given a complete bipartite graph $G = (P \cup M, E)$, a number of trucks T, a non-negative metric distance function $d(e)$ for each $e \in E$, and a distance limit L, and the aim is to find T vertex-disjoint paths $\mathscr{P}$, each starting in P and ending in M, such that $\forall p \in \mathscr{P}, \sum_{e\in p} d(e) \leq L$, so that the total number of vertices visited by at least one path is maximized.

Empirical Solution Given the importance of having the bike-share system in a good state before the morning rush hour it is crucial to quickly produce high quality routes for the overnight rebalancing shift. To achieve this we tackled the Overnight Rebalancing Problem from an empirical perspective.

$$\max \sum_{t\in\{1,\ldots,T\}} \sum_{v\in P\cup M} z_v^t \quad \text{subject to} \quad (15)$$

$$\sum_{p\in P} r^t_{(d_{start},p)} = 1 \qquad \forall t \in \{1...T\} \quad (16)$$

$$\sum_{m\in M} r^t_{(m,d_{end})} = 1 \qquad \forall t \in \{1...T\} \quad (17)$$

$$\sum_{u\in\mathcal{N}(v)} r^t_{(u,v)} = \sum_{u\in\mathcal{N}(v)} r^t_{(v,u)} \qquad \forall v \in P\cup M, \forall t \in \{1...T\} \quad (18)$$

$$\sum_{e\in E} r_e^t \cdot d(e) \le L \qquad \forall t \in \{1...T\} \quad (19)$$

$$\sum_{t\in\{1,\ldots,T\}} z_v^t \le 1 \qquad \forall v \in P\cup M \quad (20)$$

$$\sum_{u\in P\cup M} r^t_{(u,v)} = z_v^t \qquad \forall v \in P\cup M, \forall t \in \{1...T\} \quad (21)$$

$$f_e^t \le r_e^t \cdot |P\cup M| \qquad \forall e \in \bar{E}, \forall t \in \{1...T\} \quad (22)$$

$$\sum_{u\in\mathcal{N}(v)} f^t_{(u,v)} = \sum_{u\in\mathcal{N}(v)} f^t_{(v,u)} + z_v^t \qquad \forall v \in P\cup M, \forall t \in \{1...T\} \quad (23)$$

$$r^t_{(u,v)} \in \{0,1\} \qquad \forall (u,v) \in \bar{E}, \forall t \in \{1...T\} \quad (24)$$

$$f^t_{(u,v)} \in \{0...|P\cup M|\} \qquad \forall (u,v) \in \bar{E}, v\neq d_{end}, \forall t \in \{1...T\} \quad (25)$$

$$z_v^t \in \{0,1\} \qquad \forall v \in P\cup M, \forall t \in \{1...T\} \quad (26)$$

One approach to solving the Overnight Rebalancing Problem is to formulate it as an integer program; we give a formulation for which standard IP software typically computes high-quality solutions for modest-sized inputs within reasonable time bounds. Given the input graph $G = (P \cup M, E)$, we construct an augmented (directed) graph $\bar{G}$: we start with the input bipartite graph, bidirecting its edges, and add a start *depot* vertex d_{start} as well as a finish *depot* d_{end}. In addition to two directed copies of each edge $(u,v) \in E$, there is an edge from d_{start} to each vertex in P and an edge from each vertex in M to d_{end}. Let $\bar{E}$ denote the augmented set of edges, and for each $u \in P \cup M$, we let $\mathcal{N}(u)$ denote the set of vertices v for which there exists an edge $(u,v) \in \bar{E}$. In constructing an integer programming formulation, there will be three type of integer variables. First, there are 0-1 variables z_v^t that indicate whether the truck $t \in \{1,\ldots,T\}$ rebalances node v. We also have a set of 0-1 variables $r^t_{(u,v)}$ that indicate whether the edge (u,v) is on the route traversed by truck t, $t = 1,\ldots,T$. Hence, we have flow conservation constraints (16), (17), and (18), which ensure, respectively, that the path starts at a node in P, ends at a node in M, and that whenever the path enters a node, it must exit that node as well. Similarly, it is natural to have the length constraint (19) to upper bound the length of the path, and disjointness constraint (20) to ensure that at most one truck rebalances a given node v. Finally, it is clear that the node v is visited by t, if there is some edge $e = (u,v)$ for which $r_e^t = 1$; hence, we get the constraints (21).

However, if one considers the feasible solutions to just these constraints, then it is easy to forget that a feasible 0-1 solution for r^t might not correspond simply to a path, but to a path plus a collection of cycles. The role of the final set of variables is to enforce that the only nodes that are serviced by t are those nodes on the path indicated by r. For each edge $e = (u,v) \in \bar{E}$, where $u \in d_{start} \cup P \cup M$, and $v \in P \cup M$, the integer variable f_e^t counts the number of nodes in $P \cup M$ yet to be traversed in its path (and so, for example, if the path for truck 1 rebalances 8 nodes, starting node $u \in P$, then $f^1_{(d_{start},u)} = 8$). First, each variable f_e^t is positive (and of course is at most $|P \cup M|$) only when r_e^t is 1; this is enforced by constraints (22). Finally, if the count entering node v is ℓ, then the count exiting it $\ell - 1$ (provided the truck traverses node v); this is exactly captured by the constraint (23). And notice the effect of this constraint on a potential cycle selected by the variables r; its corresponding f value must decrease by 1 for each edge traversed in the cycle, but clearly this is impossible. Hence, this additional set of flow variables and constraints preclude the possibility of selecting cycles.

Heuristic Approach As the number of trucks increases, the time it takes to solve the IP increases. From experimental results shown below, one truck is solvable by the IP, whereas larger number of trucks require more than the modest time limit given to the solver. This leads us to investigate a heuristic approach to the problem, specifically a greedy algorithm. In this greedy algorithm, we repeatedly solve the IP for one truck, and then remove those vertices from the graph and solve again. Removing the vertices covered by the route is valid, since the bipartite graph is complete, and once chosen no other truck will be able to use the removed vertices.

Framing this optimization problem as a covering problem allows us to analyze properties of the greedy heuristic. In this case we are choosing a subset from a ground set of all possible truck routes, paths starting in P, ending in M of distance at most L, to cover another set, the set of all vertices. Given a subset of truck routes, the number of vertices covered is equal to twice the number of (p,m) pairs covered by trucks, where each p and each m can appear in at most one pair. To compute this, consider ordering the truck routes and take all pairs defined by the first route. For all subsequent routes, if a vertex on the route has already been visited we shortcut the pair it belongs to in the route and continue. We observe that this objective function is submodular (Lemma 4.1). This property allows a greedy approach, where at each step the best possible route for a truck is taken, to yield a $\left(1 - \frac{1}{e}\right)$ solution, as shown in Nemhauser, Wolsey and Fisher (Nemhauser, Wolsey, and Fisher 1978).

Typically, when maximizing a submodular function over a finite ground set, at each stage the element from the set that increases the objective function the most is added. However, in this case, the ground set is exponential in size requiring a different approach than iterating through the elements of the ground set. Given a problem instance, it is clear that solving the problem for one truck yields the best route from the ground set. For the following iterations, by removing routes we have already taken, the best route for one truck is equal to the best element from the ground set to add. Thus solving the 1-truck IP to find the best route at each stage gives us a $(1 - \frac{1}{e})$ approximation to the original problem (though albeit without any guarantee on how efficient the algorithm is).

Lemma 4.1 *The function mapping a set of paths of length at most L that start in P and end in M to the number of (p,c)*

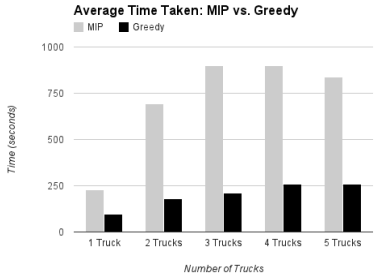

Figure 4: Average time taken by the IP and greedy approaches for different numbers of trucks on real instances.

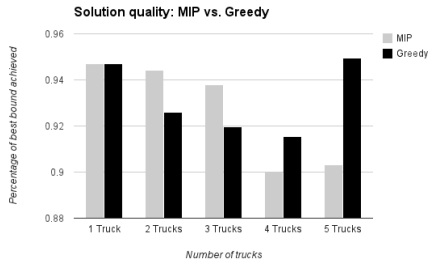

Figure 5: Solution quality found by the IP and greedy approaches for different numbers of trucks on random instances.

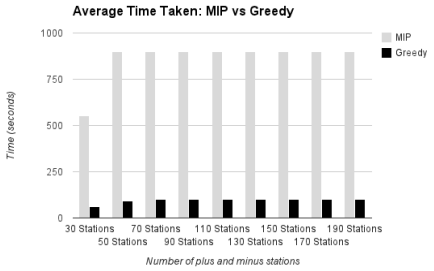

Figure 6: Average time taken by the IP and greedy approaches for different instance sizes in random instances.

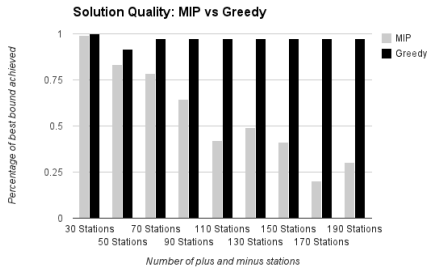

Figure 7: Solution quality found by the IP and greedy approaches for different instance sizes in random instances.

pairs (where each p and each c can appear in at most one pair) visited by at least one path is monotone submodular.

Proof Consider adding a path to the set of paths and re evaluating the objective function. There can be no decrease in the objective function as we are not removing any paths. Given two sets of paths A, B with $A \subseteq B$ we can not gain more by adding another path to B than by adding it to A as any pairs visited by A are visited in B and shortcutting that must take place in A must take place in B.

Experimental Results We tested the integer programming approaches on real world instances gathered from actual system data. We implemented the IP in Gurobi (Gurobi Optimization 2014) and carried out a number of experiments. The experiments were run on Linux machines with 2 Intel x5690s running at 3.46GHZ, a 128GB SSD and 96GB RAM. To generate the real world instances we took a series of system snapshots of the system state at the 8pm start of the overnight shift during June 2013. We used a standard plan for the system state at the start of the morning rush hour to compute the stations that make up the bipartite graph, specifying the surplus and deficit nodes P and M. The instances typically had 50 P vertices and 50 M vertices. With station location GPS information, we can compute an estimated travel distance between the stations. For each instance we vary the number of trucks available for rebalancing and analyze both the runtime taken by the solver as well as solution quality; in total, the data set contained 50 instances. We ran the IP with a 900 second cutoff and restricted the greedy to use only 300 seconds for each greedy call to the IP. The results can be seen in Figures 4 and 5. From this it is clear that as the number of trucks increases, the greedy approach produces higher quality solutions in less time than the IP. In Figure 5 we compare the solutions returned by both methods to the best bound found by the solver in 900 seconds. It is interesting to note that although one can solve the 1-truck inputs to optimality (say, with an hour of computation time), it is typically the case that this only shows the incumbent solution to be optimal.

We also conducted experiments on randomly generated instances where we fix the number of trucks at 5 (a realistic number of available trucks) and we vary the number of P and M vertices. The results of this experiment are shown in Figures 6 and 7. Again the greedy approach outperforms the MIP. The performance of the greedy solution, both in terms of time taken and solution quality allows it to be used in practice.

5 Conclusion and Future Work

In this paper, we provide a novel way of thinking about bike rebalancing in bike-share systems. Our models are motivated by operational constraints and observations gleaned from a close collaboration with bike-share operators. We provide solution methods for our models that are sufficient to provide high quality, usable solutions to real world instances from New York City as well as providing provable

guarantees on solution quality. Solutions to the mid-rush rebalancing problem are already in use in New York and the tools developed to solve the Overnight Rebalancing Problem are currently being integrated into the truck dispatching system. Our approaches provide "the overarching vision for how we like our system to look," according to Citibike director of operations Michael Pellegrino (Wald 2014).

The collaboration with New York Bike Share LLC. is ongoing: a wealth of other operational challenges remain to be tackled, such as optimizing battery replacement for stations and developing models for future system expansion. Furthermore, we are continuing to improve the solution methods for the models presented in this paper, refining both the computational results as well as improving the models through feedback from people in the field.

Acknowledgments We would like to acknowledge Shane Henderson and Lior Seeman for many helpful discussions. This research was supported in part by NSF grants CCF-0832782 and CCF-1017688.

References

Anily, S., and Bramel, J. 1999. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics (NRL)* 46(6):654–670.

Archetti, C.; Bertazzi, L.; Laporte, G.; and Speranza, M. G. 2007. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science* 41(3):382–391.

Chekuri, C.; Korula, N.; and Pál, M. 2012. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms* 8(3):23.

Chemla, D.; Meunier, F.; and Calvo, R. W. 2013. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization* 10(2):120 – 146.

Contardo, C.; Morency, C.; and L.-M. Rousseau. 2012. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Université de Montréal, Montréal, Canada.

Garca-Palomares, J. C.; Gutirrez, J.; and Latorre, M. 2012. Optimizing the location of stations in bike-sharing programs: A GIS approach. *Applied Geography* 35(1?2):235 – 246.

Gurobi Optimization, I. 2014. Gurobi optimizer reference manual.

Hernndez-Prez, H., and Gonzlez, J. J. S. 2007. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks* 50(4):258–272.

Hochbaum, D. S., and Shmoys, D. B. 1985. A best possible heuristic for the k-center problem. *Mathematics of Operations Research* 10(2):180–184.

Kaltenbrunner, A.; Meza, R.; Grivolla, J.; Codina, J.; and Banchs, R. 2010. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing* 6(4):455 – 466. Human Behavior in Ubiquitous Environments: Modeling of Human Mobility Patterns.

Larsen, J. 2013. Bike-sharing programs hit the streets in over 500 cities worldwide. *Earth Policy Institute*.

Lin, J.-R., and Yang, T.-H. 2011. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review* 47(2):284–294.

Martinez, L. M.; Caetano, L.; Eir, T.; and Cruz, F. 2012. An optimisation algorithm to establish the location of stations of a mixed fleet biking system: An application to the city of lisbon. *Procedia - Social and Behavioral Sciences* 54(0):513 – 524. Proceedings of {EWGT2012} - 15th Meeting of the {EURO} Working Group on Transportation, September 2012, Paris.

Nair, R.; Miller-Hooks, E.; Hampshire, R. C.; and Bušić, A. 2013. Large-scale vehicle sharing systems: Analysis of Velib. *International Journal of Sustainable Transportation* 7(1):85–106.

Nemhauser, G.; Wolsey, L.; and Fisher, M. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 14(1):265–294.

Rainer-Harbach, M.; Papazek, P.; Hu, B.; and Raidl, G. R. 2013. Balancing bicycle sharing systems: A variable neighborhood search approach. In Middendorf, M., and Blum, C., eds., *EvoCOP*, volume 7832 of *Lecture Notes in Computer Science*, 121–132. Springer.

Raviv, T.; Tzur, M.; and Forma, I. 2013. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics* 2(3):187–229.

Romero, J. P.; Ibeas, A.; Moura, J. L.; Benavente, J.; and Alonso, B. 2012. A simulation-optimization approach to design efficient systems of bike-sharing. *Procedia - Social and Behavioral Sciences* 54(0):646 – 655. Proceedings of {EWGT2012} - 15th Meeting of the {EURO} Working Group on Transportation, September 2012, Paris.

Schuijbroek, J.; Hampshire, R.; and van Hoeve, W.-J. 2013. Inventory rebalancing and vehicle routing in bike sharing systems. *Tepper School of Business* Paper 1491.

Shu, J.; Chou, M. C.; Liu, Q.; Teo, C.-P.; and Wang, I.-L. 2013. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research* 61(6):1346–1359.

Vansteenwegen, P.; Souffriau, W.; and Oudheusden, D. V. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209(1):1 – 10.

Vogel, P., and Mattfeld, D. C. 2011. Strategic and operational planning of bike-sharing systems by data mining - a case study. In Bse, J.; Hu, H.; Jahn, C.; Shi, X.; Stahlbock, R.; and Vo, S., eds., *ICCL*, 127–141. Springer.

Wald, C. 2014. Wheels when you need them. *Science* 345(6199):862–863.