Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems

J. Schuijbroek[a,*], R.C. Hampshire[b,1], W.-J. van Hoeve[c]

[a]*School of Industrial Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, Netherlands*
[b]*University of Michigan Transportation Research Institute, 2901 Baxter Rd, Ann Arbor, MI 48109-2150, United States of America*
[c]*Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, United States of America*

Abstract

Bike sharing systems have been installed in many cities around the world and are increasing in popularity. A major operational cost driver in these systems is rebalancing the bikes over time such that the appropriate number of bikes and open docks are available to users. We combine two aspects that have previously been handled separately in the literature: determining service level requirements at each bike sharing station, and designing (near-)optimal vehicle routes to rebalance the inventory. Since finding provably optimal solutions is practically intractable, we propose a new cluster-first route-second heuristic, in which a polynomial-size Clustering Problem simultaneously considers the service level feasibility and approximate routing costs. Extensive computational results on real-world data from Hubway (Boston, MA) and Capital Bikeshare (Washington, DC) are provided, which show that our heuristic outperforms a pure mixed-integer programming formulation and a constraint programming approach.

Keywords: routing, inventory, integer programming, constraint programming, Markov processes

*Corresponding author
Email addresses: `jasper.schuijbroek@gmail.com` (J. Schuijbroek), `hamp@umich.edu` (R.C. Hampshire), `vanhoeve@andrew.cmu.edu` (W.-J. van Hoeve)

[1]This material is based upon work partially supported by the National Science Foundation under Grant No. CMMI #1055832.

1. Introduction

Bike sharing systems are experiencing wide-spread adoption in major cities around the world, with over 1,000 active systems and more than 300 in planning or under construction (Meddin & DeMaio, 2016). In these systems, users can pickup and return bikes at designated bike sharing stations with a finite number of docks. Unfortunately, user behavior results in spatial imbalance of the bike inventory over time. The system equilibrium is often characterized by unacceptably low availability of bikes or open docks, for pickups or returns respectively (Fricker et al., 2012, p. 375).

Therefore, operators deploy a fleet of trucks to *rebalance* the bike inventory. We focus on the efficiency of these rebalancing operations, a major cost driver for operators (DeMaio, 2009, p. 50). This problem consists of two main components. First, determining the desired inventory level at each bike station, which is typically done by an analysis of historic user data. Second, designing truck routes that will perform the necessary pickups and deliveries in order to reach the target inventory levels.

In practice, operators may perform a major rebalancing operation during the night (the *static* case, assuming negligible user activity during repositioning), as well as continuous rebalancing during the daytime (the *dynamic* case). Finding (near-)optimal solutions to these routing problems may take several hours even for the most advanced optimization models available (Raviv et al., 2013; Contardo et al., 2012).

We present a rigorous cluster-first route-second heuristic for the *static* non-stationary case that combines inventory flexibility and routing decisions. These aspects have mostly been considered separately in literature (see Section 2 for a literature review). Combining routing and inventory flexibility may lead to more efficient solutions, and allows operators to implement the tradeoff between user satisfaction and the cost of rebalancing, which is often faced in practice.

Our approach relies on target inventory bounds at each bike station that represent service level requirements. In particular, we model the stochastic demand by viewing the inventory at each station as a non-stationary queuing system with finite capacity, and derive service level requirements using the transient distribution of the availability of bikes and docks (Section 3). We recognize that service level requirements can be met when the inventory is between a lower and upper bound. Any other suitable uncertainty model could be used for this purpose.

Our proposed clustering heuristic proceeds by grouping together bike stations such that each cluster is 'self-sufficient', i.e. the target inventory bounds can be satisfied by only performing within-cluster pickups and deliveries. The novelty in our approach is that we estimate the routing cost of each cluster by a new *maximum spanning star* approximation. A major benefit of the maximum spanning star approximation is that it can be formulated using a linear[2] number of constraints only. The resulting mixed-integer program (MIP) for our clustering problem can be solved optimally within seconds for realistic problem sizes. For each cluster an optimal route can be computed easily using a standard MIP formulation.

We also present a Constraint Programming (CP) formulation in Section 6, acting both as an effective (exact) solution method for smaller instances and a benchmark for our dedicated clustered routing heuristics.

We compare our approach to an exact MIP model based on existing literature, as well as our CP model. Computational experiments (Section 7) using data from Hubway (Boston, MA) and Capital Bikeshare (Washington, DC) show that our heuristics strongly outperform the classical MIP model. Within seconds, we identify a feasible solution with a reasonable optimality gap. In a minute, we find better solutions for almost all instances than the best MIP solution after 2 hours. Moreover, our dedicated Clustered MIP heuristic outperforms Constraint Programming on larger instances.

2. Related Work

The study of bike sharing systems is increasing in popularity. DeMaio (2009) and Shaheen et al. (2010) provide a history of bicycle sharing, starting with the first generation 'white bikes' in Amsterdam as early as 1965. From 1995 onwards, the third generation IT-based systems incorporate 'advanced technologies for bicycle reservations, pickup, drop-off, and information tracking' (Shaheen et al., 2010, p. 7). For the interested reader Laporte et al. (2015) provides a comprehensive survey of the vehicle/bike sharing literature. We identify four research substreams in bike sharing literature: strategic design, demand analysis, service level analysis, and rebalancing operations.

[2]In the number of bike stations

Strategic design

Since most major cities have planned or considered implementation of bike sharing systems, and existing systems are often expanded, several studies present models dedicated to strategic design. Dell'Olio et al. (2011) develop a comprehensive methodology for implementation, from estimating potential demand to optimizing locations. Martinez et al. (2012) and Prem Kumar & Bierlaire (2012) present MIP models for the location problem. Lin & Yang (2011) create a model trading off the interests of both users and investors. We note that none of the studies include a notion of expected inventory imbalance costs.

Demand analysis

The purpose of demand analysis is twofold: forecasting future demand (e.g. for service level requirements) and understanding the explaining factors for managerial decision making. Kaltenbrunner et al. (2010) predict the system inventory state and suggest making such information available to users. Froehlich & Oliver (2008); Borgnat & Abry (2009); Borgnat et al. (2011), and Lathia et al. (2012) identify a temporal demand pattern and forecast the number of rentals. Hampshire et al. (2013) seek land use and socio-economic factors that explain the use of bike sharing systems. Vogel et al. (2011) construct clusters of stations with similar demand patterns. These studies could provide insights that are helpful in improving the service level requirements developed in this paper.

Service level analysis

Several studies focus on service levels in bike sharing systems. We recall that service level requirements are typically two-sided: for available bikes and docks. Most notably, Nair & Miller-Hooks (2011) and Nair et al. (2013) decompose system-wide reliability into a set of dual-bounded chance constraints for each station. We adapt their dual-bounded service level constraints, but use a more realistic Markov chain to model the station inventory over time, as opposed to observing the total net demand (see Section 3). Raviv & Kolka (2013) present a queuing system with finite capacity to model expected user dissatisfaction at each station, similar to our approach. Leurent (2012) models bike sharing stations as a dual Markovian waiting system, but contrary to their assumption of waiting customers, we assume immediate lost sales.

George & Xia (2011) develop a method to determine fleet sizing given a desired service level. This is accomplished by modeling the number of bikes at each station and their movements as a closed queueing network. Their results follow from a steady state analysis of the queueing network. The work does not take the capacity of stations into account, and their service level is measured in terms of the availability of bikes as opposed to unmet demand for pickups/returns. Our paper addresses these two limitations in the non-steady state (transient) regime. Additionally, we consider the related routing problem, which George & Xia do not consider.

Rebalancing operations

Using mean field analysis, Fricker et al. (2012) conclude that the equilibrium system performance collapses under heterogeneity of user behavior and that a pressing need for rebalancing operations exists. Vogel & Mattfeld (2010) motivate rebalancing activities using a system dynamics model. Shu et al. (2013) estimate rebalancing operations can lead to an additional 15-20% of trips supported system-wide.

We identify two modes of rebalancing: providing user incentives and deploying a truck fleet. While Fricker & Gast (2014) and Waserhole & Jost (2012) develop a pricing strategy, and some active systems have already implemented incentive schemes (e.g. *V+* for Vélib', see Fricker & Gast, 2014), we observe that every bike sharing system still operates a vehicle fleet for rebalancing. Therefore, the underlying vehicle routing problem has received most attention.

Most studies use an exact target inventory for each station, which implies their routing problem is more closely related to the One-Commodity Pickup-and-Delivery VRP (Hernández-Pérez & Salazar-González, 2003) than ours, which adds inventory flexibility. Routing costs to attain exact target inventories will always be larger than or equal to the costs strictly necessary to maintain appropriate service levels (see Section 3). We note that our models can be parameterized to solve the target inventory problem.

Erdoğan et al. (2013) consider the pickup and delivery redistribution problem with *one vehicle* where the target inventory at each station must lie in an interval. The intervals are input to their problem. Our problem is a natural extension of this setting in that we consider multiple vehicles and we determine the target inventory intervals. Chemla et al. (2013) present a branch-and-cut algorithm for the static single-vehicle problem, with results on instances of up to 100 stations. Approximation algorithms for the same problem are given by Benchimol et al. (2011). Erdoğan et al. (2015) develop an exact method to calculate the optimal route for the static *single-vehicle* rebalancing problem. Instances with up to 60 stations can be solved to optimality within 2 hours of computing time.

Raviv et al. (2013) present an alternative static approach: expected system-wide user dissatisfaction is minimized subject to a time limit. Their arc-, time-, and sequence-indexed MIP models are intractable for systems of reasonable size, however we use these models in Section 4 to present the Bike Sharing Rebalancing Problem in its pure MIP form.

Di Gaspero et al. (2013b) proposed a constraint programming formulation for the vehicle routing problem that minimizes user dissatisfaction based on the total deviation from fixed target levels for each bike station. Then Di Gaspero et al. (2015) consider the design of optimal tours and operating instructions for relocating bikes among stations. The authors propose a constraint programming models with Large Neighborhood Search (LNS). However, their model formulation considers only the routing problem and takes the target inventory levels as given.

Rainer-Harbach et al. (2014) also consider the joint problem of inventory balancing and vehicle routing, and propose dedicated (meta-)heuristic approaches. They first present several construction heuristics, which can be further improved by applying problem-specific local search moves, including the insertion or removal of stations in a route, or the swap of stations within routes. Care must be taken that the proposed moves, as well as the construction heuristics, are feasible. In addition, the objective is to minimize the weighted sum of a) the deviation from a given target inventory level for each station, b) the number of (un)loading operations, and c) total work time. The authors show that their heuristics are able to find high-quality solutions in a reasonable time.

Di Gaspero et al. (2013a) combine a CP optimization model with Ant Colony Optimization to solve a similar problem as Rainer-Harbach et al. (2014), again minimizing a weighted sum of the deviation from target inventory levels and total work time. Their CP model uses a direct 'successor' formulation with integer variables, whereas the CP model we propose in Section 6 is based on constraint-based scheduling.

Kloimüllner et al. (2015) develop a cluster-first route-second approach. Their objective function is to maximize the number of stations visited over a fixed time window. Their algorithm is exact and utilizes a Benders composition. However, their routes must alternate between pickup and delivery stations. This is a major drawback of their approach.

The paper by Forma et al. (2015) proposes 3-step heuristic for the static repositioning problem. Step 1 clusters stations that are close to each other and have compatible demand structures. Step 2 routes the vehicles through the clusters and makes heuristic inventory targets for the stations in each cluster. Step 3 uses the results of steps 1 and 2 to determine both the routing within the clusters and the inventory levels for each station. Their clusters are constructed to reduce the size of the network, and not to decompose the problem into many single vehicle routing problems. Additionally, their routing step allows vehicles to travel between clusters. The proposed 3 step method is successfully applied to several real life instances. However, the authors provide no theoretical bounds or guarantees of the proposed method.

The PhD dissertation of O'Mahony (2015) at Cornell University builds on our paper. The models developed in the dissertation are currently implemented in the operations of Citi Bike Share in New Year City. The author departs from our model in that he does not compute upper and lower inventory levels. By contrast, he computes a convex cost function for each starting inventory level. The corresponding routing problem is to minimize the service level 'costs' over a fixed time interval. He argues that the minimal makespan is often much longer than the time interval of a typical staff work shift.

3. Service Level Requirements

In general, inventory rebalancing efforts are made in order to improve customer service. Contrary to traditional inventory theory, where the service level increases as inventory increases, bike sharing systems are subject to a *net demand process*, with an empty station preventing users from picking up bikes *and* a full station preventing returns. For the remainder of the paper, we let $\mathcal{S}$ represent the set of bike sharing stations. Let C_i denote the capacity (i.e. number of docks) of station $i \in \mathcal{S}$.

Net demand implies that after a pickup and a return occur, the inventory is unchanged. Previous studies (Nair et al., 2013) observe the *total* net demand (pickups minus returns during observation period) at each station $i \in \mathcal{S}$. However, we note that while observing a net demand of zero, we often still need bikes or docks. We view the net demand as a stochastic process which needs to be satisfied during the entire observation period, not only at the end.

3.1. Service Level Definition

We implement a *type 2* service level: the *fraction of demands satisfied directly* should be larger than β_i^- for pickups and larger than β_i^+ for returns. We assume no backorders, i.e., the effect of waiting customers is negligible.

Definition 1. The service level requirements at station $i \in \mathcal{S}$ are

$$\frac{\text{E[Satisfied bike pickup demands]}}{\text{E[Total bike pickup demands]}} \geq \beta_i^-$$

$$\frac{\text{E[Satisfied bike return demands]}}{\text{E[Total bike return demands]}} \geq \beta_i^+$$

for given $\beta_i^-, \beta_i^+ \in [0, 1]$.

3.2. Markov Chain Formulation

We assume that user behavior during the observation period is non-stationary, i.e. rates depend on time. This allows us to implement the model over multiple time periods (e.g. morning 6–9AM or evening 3–6PM). The time-dependent rates allow the model to capture the demand changes observed in practice. Our goal is to select an initial number of bikes that achieve the service level requirements over the observation period for each station. While some users arrive in groups (compound Poisson process), we assume this effect is negligible.

The inventory at station $i \in \mathcal{S}$ can be modeled as an $M_t/M_t/1/K$ queuing system (Kendall, 1953), with the number of customers in the queue representing the inventory. This implies that the customer inter-arrival times (for bike returns) and service times (i.e. inter-arrival times for bike pickups) are exponentially distributed with time dependent rates $\lambda_i(t)$ and $\mu_i(t)$, respectively, at each station $i \in \mathcal{S}$. There are $K = C_i$ waiting spaces in the system for station $i \in \mathcal{S}$. The stationary $M/M/1/K$ queue is well-studied and closed-form expressions for the transient probabilities given a starting state are available. In the non-stationary case, $M_t/M_t/1/K$, the transient probabilities given a starting state are solutions to the Kolmogorov forward equations (Morse, 1958).

This method computes the service levels for each station separately. The approach does not account for the interactions of supply and demand between stations. To properly account for these interactions, a state dependent model of the 'lost sales' and 'lost returns' is needed. In order to jointly calculate service levels for all stations the resulting model would likely be a time dependent queueing network with a complex abandonment and routing structure. Such a model is beyond the scope of this study.

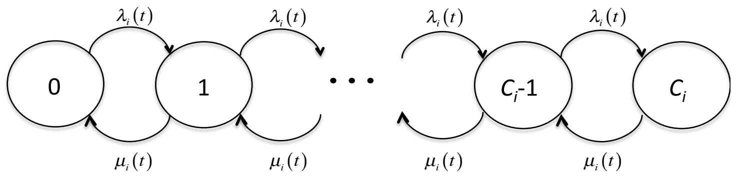

Figure 1: Markov chain for the inventory $S_i(t)$ at station $i \in \mathcal{S}$.

Note. Users arrive with rate $\lambda_i(t)$ to return bikes and with rate $\mu_i(t)$ to pickup bikes.

Denote by $\{S_i(t) : t \geq 0\}$ the stochastic process on state space $\{0, \dots, C_i\}$ representing the inventory of station $i \in \mathcal{S}$ at time $t \geq 0$. Define $p_i(s, \sigma, t) \equiv \Pr(S_i(t) = \sigma \mid S_i(0) = s)$, the transient probability that the inventory at station $i \in \mathcal{S}$ equals $\sigma \in \{0, \dots, C_i\}$ at time $t \geq 0$ given starting inventory $s \in \{0, \dots, C_i\}$.

Then, in order to meet both the pickup and return service level from Definition 1 during the observation period $[0, T]$, we calculate the expected values:

$$\begin{aligned}\frac{\text{E[Satisfied bike pickup demands]}}{\text{E[Total bike pickup demands]}} &= \frac{\int_0^T \mu_i(t)\,(1 - p_i(s, 0, t))\;\mathrm{d}t}{\int_0^T \mu_i(t) dt} \\ &\equiv g_i(s, 0, T).\end{aligned}$$

and similarly

$$\begin{aligned}\frac{\text{E[Satisfied bike return demands]}}{\text{E[Total bike return demands]}} &= \frac{\int_0^T \lambda_i(t)\,(1 - p_i(s, C_i, t))\;\mathrm{d}t}{\int_0^T \lambda_i(t) dt} \\ &\equiv g_i(s, C_i, T).\end{aligned}$$

The transient probabilities solve the Kolmogorov forward equations

$$\dot{p}_i(s, 0, t) = \mu_i(t) \cdot p_i(s, 1, t) - \lambda_i(t) \cdot p_i(s, 0, t)$$

$$\begin{aligned}\dot{p}_i(s,\sigma,t) &= \mu_i(t)\cdot p_i(s,\sigma+1,t) - \lambda_i(t)\cdot p_i(s,\sigma-1,t) \quad \sigma = 1,\dots,C_i-1\\ \dot{p}_i(s,C_i,t) &= \lambda_i(t)\cdot p_i(s,C_i-1,t) - \mu_i(t)\cdot p_i(s,C_i,t)\end{aligned}$$

Intuitively, as the starting inventory increases, the bike pickup service level increases and the bike return service level decreases. Confirming this intuition, the proof of Theorem 1 in Raviv & Kolka (2013, p. 9) yields that the bike pickup and bike return service level are respectively increasing and decreasing in the starting inventory, which gives us Lemma 1.

Lemma 1. *Denote by $s_{i,T}$ the station inventory of station $i \in \mathcal{S}$ after rebalancing operations for an observation period $[0,T]$. To meet the service level requirements from Definition 1 at station $i \in \mathcal{S}$, it must hold that $s_{i,T} \geq s_{i,T}^{\min}$ and $s_{i,T} \leq s_{i,T}^{\max}$ with*

$$\begin{aligned} s_{i,T}^{\min} &= \min\left\{s \in \{0,\dots,C_i\} : g_i(s,0,T) \geq \beta_i^-\right\}\\ s_{i,T}^{\max} &= \max\left\{s \in \{0,\dots,C_i\} : g_i(s,C_i,T) \geq \beta_i^+\right\}.\end{aligned}$$

Hence, for an observation period $[0,T]$ the vehicles should rebalance the starting inventory s_i^0 such that $s_{i,T}^{\min} \leq s_{i,T} \leq s_{i,T}^{\max}$ for each station $i \in \mathcal{S}$, in order to meet the service levels β_i^-, β_i^+. Denote by $\mathcal{S}_0 = \{i \in \mathcal{S} : s_{i,T}^{\min} \leq s_i^0 \leq s_{i,T}^{\max}\}$ the set of *self-sufficient* stations which satisfy the service level requirements over time period [0,T] with their starting inventory.

Note that the service level requirements could theoretically be *infeasible* over a given observation period. We identify three types of infeasibility:

- $s_{i,T}^{\max} < s_{i,T}^{\min}$ requires the operator to prioritize the service level requirement for either pickups or returns at station $i \in \mathcal{S}$, or choose a weighted average.
- $g_i(C_i,0,T) < \beta_i^-$ or $g_i(0,C_i,T) < \beta_i^+$ implies the inventory always violates (one of) the service level requirements. This requires the operator to choose the best possible inventory bounds.
- $\sum_{i\in\mathcal{S}} s_i^0 < \sum_{i\in\mathcal{S}} s_{i,T}^{\min}$ or $\sum_{i\in\mathcal{S}} s_i^0 > \sum_{i\in\mathcal{S}} s_{i,T}^{\max}$ implies a system-wide shortage or excess (ignoring vehicle inventory and capacity for the sake of simplicity).

All types of infeasibility would require the operator to take alternative measures in case service level problems persist, e.g., increase station capacity,

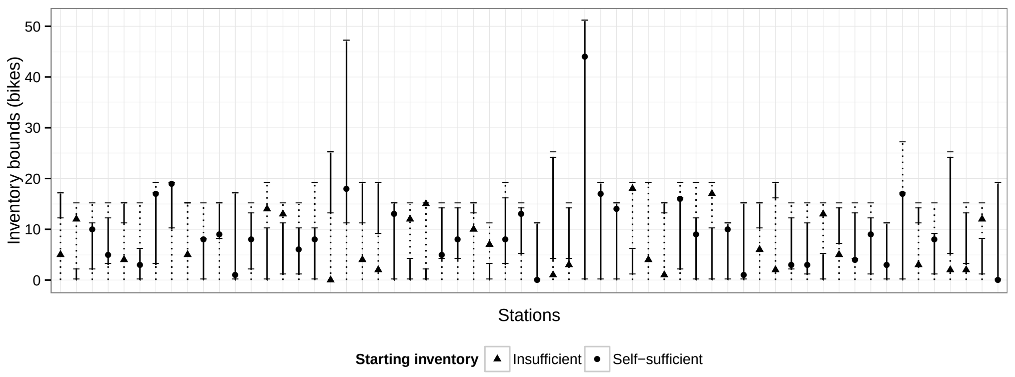

Figure 2: Example of Service Level Requirements for Hubway (Boston, MA).

Note. Observation period 6–9AM ($T = 3$) on weekdays with $\beta_i^- = \beta_i^+ = 85\%$. The range $[s_i^{\min}, s_i^{\max}]$ is displayed solid with $[0, C_i]$ displayed dotted. For reference, we show the starting inventory s_i^0 for each station as a solid marker, based on a snapshot of the system taken at 6AM on June 26st 2012.

influence user behavior, or introduce or remove bikes from the system. We introduce $s_i^{\min}$ and $s_i^{\max}$ for the inventory bounds adjusted for infeasibility.

Using Lemma 1, we are able to efficiently calculate the service level requirements at each station. We solve the Kolmogorov forward equations for the transient state probabilities using the fourth-order Runge–Kutta method. We compute the service levels over increasingly larger observation periods, i.e. 6–7AM, 6–8AM, 6–9AM, etc. The number of infeasible stations typically increases with the length of the observation period. We select the observation period that has up to 10–15% infeasible stations, since for several stations infeasibility cannot be prevented, e.g. due to limited station capacity. For infeasible stations, where necessary we prioritize pickups over returns, i.e. $s_i^{\max} = \max\{s_{i,T}^{\max}, s_{i,T}^{\min}\}$, and/or use $s_i^{\min} = 0$ and $s_i^{\max} = C_i$.

Below is an example using real data from the bike sharing system Hubway in Boston, MA.

Example 1. For this example, we use trip data provided by Hubway (Boston, MA) for the 60 stations that were active between November 1st 2011 and May 31st 2012. In Appendix A, we give a detailed overview of the data sources used for our examples and computational results. The observation period is 6–9AM ($T = 3$) on weekdays, for which we have 82 observations.

We estimate $\lambda_i(t)$ and $\mu_i(t)$ by the mean (Maximum Likelihood Estimator for Poisson variables) number of returns and pickups, respectively, per 15-minute interval during the observation period.

These observations suffer from the 'lost sales' bias, i.e. unmet demand is not recorded. Moreover, some lost sales are endogenous to the rebalancing operations, e.g. removing bikes which due to the stochasticity of demand turn out to be necessary. We note that these effects are stochastically complex to capture and are out of scope for this study.

We require a $\beta_i^- = \beta_i^+ = 85\%$ service level at each station $i \in \mathcal{S}$ and quickly calculate $s_i^{\min}$ and $s_i^{\max}$ using Lemma 1. Step 1 is to set $s = 0$ and solve the Kolmogorov forward equations using the fourth-order Runge–Kutta method to obtain $g_i(s, 0, T)$. If $g_i(s, 0, T) \geq \beta_i^-$, then the lower bound $s_i^{\min} = s$, otherwise increment $s = s + 1$ and repeat Step 1. If $g_i(C_i, 0, T) < \beta_i^-$, then the station i is considered infeasible. The analogous procedure leads to the upper inventory bound $s_i^{\max}$. We adjust the bounds for 10 infeasible stations (out of 60) as described earlier.

The service level requirements for each station are depicted as bounds in Figure 2. We observe that there is a lot of flexibility in the target inventory, with most stations having relatively wide service level bounds. For some of the stations, $s_i^{\min}$ is relatively high, but this is paired with a higher than average $s_i^{\max}$, in anticipation of bike pickups clearing sufficient docks for returns.

4. Routing Problem

Recall that vehicles should rebalance the starting inventory s_i^0 such that $s_i^{\min} \leq s_i \leq s_i^{\max}$ for each station $i \in \mathcal{S}$, in order to meet the service level requirements. We define the Routing Problem, a pure MIP approach to the static Bike Sharing Rebalancing Problem, inspired by Raviv et al. (2013). The bike sharing system is represented as a complete directed graph with vertex set $\mathcal{S}$ and distances (routing costs) $d_{i,j}$ for all $i, j \in \mathcal{S}$. We note that our model is *both* arc- and sequence-indexed, which increases the size, but yields much stronger relaxations than the sequence-indexed model.

Several objectives could be applied to this routing problem, e.g. minimizing the total routing time or minimizing user dissatisfaction. However, from conversations with bike sharing operators and official sources (New York City Bike Share, 2013) we learn that service levels are often a (contractual) necessity and time for *static* overnight rebalancing is by definition limited.

Therefore, our approach is to meet the service level requirements as soon as possible (a minimize *makespan* objective), rather than maximizing service given a time limit (e.g. Raviv et al., 2013).

Note that $\mathcal{N} \equiv \mathcal{S} \cup \{0\}$ extends the station set with an artificial vertex 0 for purposes mentioned below. We use a set $\mathcal{T} = \{1, \ldots, T\}$ for sequence indexing. We introduce binary decision variables $x_{i,j,t,v}$ to indicate whether vehicle $v \in \mathcal{V}$ traverses arc (i, j) in time step $t \in \mathcal{T}$. Decision variables $y^-_{i,t,v}, y^+_{i,t,v}$ indicate bike pickup or delivery, respectively, by vehicle $v \in \mathcal{V}$. As vehicle operators spend a large amount of time (un)loading bikes, we introduce d^- and d^+ for the routing costs of picking up and delivering *one* bike, respectively. We use q^0_v and Q_v for the initial inventory and capacity of vehicle $v \in \mathcal{V}$, respectively. The formulation then becomes:

$$\text{minimize } H \tag{P1}$$

s.t.

$$s^0_i + \sum_{\substack{t\in\mathcal{T}\\ v\in\mathcal{V}}} (y^+_{i,t,v} - y^-_{i,t,v}) \geq s^{\min}_i \qquad \forall i \in \mathcal{S} \tag{1}$$

$$s^0_i + \sum_{\substack{t\in\mathcal{T}\\ v\in\mathcal{V}}} (y^+_{i,t,v} - y^-_{i,t,v}) \leq s^{\max}_i \qquad \forall i \in \mathcal{S} \tag{2}$$

$$\sum_{\substack{i\in\mathcal{S}\\ j\in\mathcal{N}}} x_{i,j,1,v} \leq 1 \qquad \forall v \in \mathcal{V} \tag{3}$$

$$\sum_{j\in\mathcal{N}} x_{i,j,t,v} \leq \sum_{j\in\mathcal{S}} x_{j,i,t-1,v} \qquad \forall i \in \mathcal{S}, t \in \mathcal{T} \setminus \{1\}, v \in \mathcal{V} \tag{4}$$

$$\sum_{\substack{i\in\mathcal{S}\\ t\in\mathcal{T}\\ v\in\mathcal{V}}} x_{i,i,t,v} = 0 \tag{5}$$

$$y^-_{i,t,v} \leq Q_v \sum_{j\in\mathcal{N}} x_{i,j,t,v} \qquad \forall i \in \mathcal{S}, t \in \mathcal{T}, v \in \mathcal{V} \tag{6}$$

$$y^+_{i,t,v} \leq Q_v \sum_{j\in\mathcal{S}} x_{j,i,t,v} \qquad \forall i \in \mathcal{S}, t \in \mathcal{T}, v \in \mathcal{V} \tag{7}$$

$$\sum_{\substack{t\in\mathcal{T}\\ v\in\mathcal{V}}} y^-_{i,t,v} \leq s^0_i \qquad \forall i \in \mathcal{S} \tag{8}$$

$$\sum_{\substack{t\in\mathcal{T}\\v\in\mathcal{V}}} y^+_{i,t,v} \leq C_i - s^0_i \qquad \forall i \in \mathcal{S} \quad (9)$$

$$q^0_v + \sum_{\substack{i\in\mathcal{S}\\ \tilde{t}\in\mathcal{T}:\tilde{t}\leq t}} (y^-_{i,t,v} - y^+_{i,t,v}) \geq 0 \qquad \forall t \in \mathcal{T}, v \in \mathcal{V} \quad (10)$$

$$q^0_v + \sum_{\substack{i\in\mathcal{S}\\ \tilde{t}\in\mathcal{T}:\tilde{t}\leq t}} (y^-_{i,t,v} - y^+_{i,t,v}) \leq Q_v \qquad \forall t \in \mathcal{T}, v \in \mathcal{V} \quad (11)$$

$$h_v = \sum_{\substack{i,j\in\mathcal{S}\\t\in\mathcal{T}}} d_{i,j} x_{i,j,t,v} + \sum_{\substack{i\in\mathcal{S}\\t\in\mathcal{T}}} d^- y^-_{i,t,v} + \sum_{\substack{i\in\mathcal{S}\\t\in\mathcal{T}}} d^+ y^+_{i,t,v} \qquad \forall v \in \mathcal{V} \quad (12)$$

$$H \geq h_v \qquad \forall v \in \mathcal{V} \quad (13)$$

$$x_{i,j,t,v} \in \{0,1\} \qquad \forall i \in \mathcal{S}, j \in \mathcal{N}, t \in \mathcal{T}, v \in \mathcal{V}$$

$$y^-_{i,t,v}, y^+_{i,t,v} \in \mathbb{N}_0 \qquad \forall i \in \mathcal{S}, t \in \mathcal{T}, v \in \mathcal{V}$$

$$H, h_v \geq 0 \qquad \forall v \in \mathcal{V}$$

Constraints (1)–(2) impose the service level requirements from Lemma 1. Constraints (3)–(7) take care of vehicle routing. Constraints (3) imply that each vehicle starts at most one route. Constraints (4) take care of flow conservation. Constraints (5) prevent dwelling. Constraints (6)–(7) ensure pickup or delivery can only take place when leaving from or arriving at a station, respectively. Note that $\mathcal{N}$ allows picking up bikes at the final stop without incurring routing costs. Constraints (8)–(9) limit the amount of transshipments, because of the model's inability to track station inventory over time (cf. Raviv et al., 2013, p. 10). Constraints (10)–(11) ensure that the vehicle inventory remains non-negative and within vehicle capacity at all times. Constraints (12) define the routing costs for each vehicle by adding up the total distance, loading, and unloading costs. Finally, constraints (13) linearize the objective $H = \max_{v\in\mathcal{V}} h_v$.

Denote by $H^*(\mathcal{S}, \mathcal{V})$ the optimal solution obtained by solving (P1) for station set $\mathcal{S}$ and vehicle set $\mathcal{V}$. Solving (P1) gives an optimal solution for the Bike Sharing Rebalancing Problem (apart from the limited transshipments).

However, we observe that (P1) is practically intractable for realistic station sets with $|\mathcal{S}| \geq 50$ and vehicle fleets with $|\mathcal{V}| \geq 3$. Therefore, we introduce Clustered Routing heuristics in Section 5 and a Constraint Programming heuristic in Section 6 as alternatives to find high quality solutions in a short amount of time.

5. Clustered Routing

We next formulate a Clustering Problem to decompose the Routing Problem (P1) into separate *single-vehicle* Routing Problems, thereby reducing combinatorial complexity.

A feasible clustering solution assigns disjoint clusters of stations $\mathcal{S}_v \subseteq \mathcal{S}$ to vehicles $v \in \mathcal{V}$ such that the service level requirements can be satisfied using *only* within-cluster vehicle routing. Implementing these feasibility constraints in existing clustering algorithms, e.g. the Fisher-Jaikumar algorithm (Fisher & Jaikumar, 1981), is non-trivial (see Section 5.3). Therefore, we propose to formulate the Clustering Problem as an extended Set Partitioning Problem. The core of the model is a set of binary decision variables:

$$z_{i,v} = \begin{cases} 1 & \text{if station } i \in \mathcal{S} \text{ is assigned to vehicle } v \in \mathcal{V}, \\ 0 & \text{otherwise.} \end{cases}$$

These variables assign a cluster of stations $\mathcal{S}_v = \{i \in \mathcal{S} : z_{i,v} = 1\}$ to each vehicle $v \in \mathcal{V}$.

The objective of the Clustering Problem is to find a feasible solution while minimizing makespan $H = \max_{v \in \mathcal{V}} h_v$, i.e. rebalance the system as soon as possible, likely by dividing the workload between vehicles. Optimally, stations are clustered with known exact routing costs for any (feasible) combination of stations $\mathcal{S}_v$. However, the computational complexity (see Section 4) requires us to use approximations to estimate the routing costs. Therefore, we are interested in non-algorithmic routing costs approximations that correlate highly with the exact vehicle routing costs within a cluster.

5.1. Routing distance approximation

First, we focus on the distance component of approximating the routing costs. Until Section 5.2 we ignore (un)loading costs, i.e. $d^- = d^+ = 0$, for notational convenience. It is not straightforward to approximate the optimal routing distance for a cluster, because feasibility constraints on the vehicle route may require revisits. However, assuming that all stations in $\mathcal{S}_v$ need to be visited, the within-cluster routing costs $H^*(\mathcal{S}_v, \{v\})$ are bounded from below by the optimal solution of a Traveling Salesman Problem with an added artificial zero-distance depot (to model the open tour for rolling horizon implementation). If we impose the triangle inequality, then this lower bound equals the length of the shortest Hamiltonian path over $\mathcal{S}_v$.

TSP approximations are widely studied, see e.g. Laporte (1992). However, exponential approximations do not scale well in MIPs and, as mentioned, we refrain from algorithmic approximations because of the feasibility constraints. For example, the MIP formulation of the Held-Karp relaxation (Held & Karp, 1970; Charikar et al., 2004) would require constraints for each non-empty subset of $\mathcal{S}$. Furthermore, the polynomial-size Assignment Problem relaxation (Dantzig et al., 1954) has limited applicability, because it does not satisfy *monotonicity*, i.e., adding stations to a cluster may actually decrease the routing costs approximation. Since sub-tours are not eliminated, our experimentation with the Assignment Problem resulted in the undesirable assignment of geographically separated groups of stations to the same cluster.

Instead, we introduce the Maximum Spanning Star approximation[3]. Denote by $\text{SPS}_i(\mathcal{S}_v) = \sum_{j \in \mathcal{S}_v} d_{i,j}$ the cost of the spanning star (spanning tree with depth one) of $\mathcal{S}_v$ rooted at station $i \in \mathcal{S}_v$. The routing costs are approximated by the maximum-cost spanning star $\text{MAXSPS}(\mathcal{S}_v) = \max_{i \in \mathcal{S}_v} \text{SPS}_i(\mathcal{S}_v)$. In Section 5.3 we show that the Maximum Spanning Star can be implemented using a polynomial number of binary assignment variables and constraints. Moreover, the Maximum Spanning Star is an upper bound on the shortest Hamiltonian path over the cluster. Most importantly, the Maximum Spanning Star satisfies monotonicity, such that stations are only assigned to a cluster if this is necessary for feasibility of the service level requirements.

We use the monotonicity and upper bound of the Maximum Spanning Star approximation, to motivate our intuition that $\text{MAXSPS}(\mathcal{S}_v)$ and $H^*(\mathcal{S}_v, \{v\})$ correlate. Namely, both $H^*(\mathcal{S}_v, \{v\})$ and $\text{MAXSPS}(\mathcal{S}_v)$ are bounded from below by the shortest Hamiltonian path over $\mathcal{S}_v$, given that the Maximum Spanning Star satisfies monotonicity (which ensures all stations are visited).

Lemma 2. *The Maximum Spanning Star approximation satisfies monotonic-*

[3]Note that Wu et al. (1998, p. 2) introduce the algorithmic minimum k-star approximation.

ity:

$$\text{MAXSPS}(\mathcal{S}_v \cup \{i\}) \geq \text{MAXSPS}(\mathcal{S}_v).$$

Denote by $\text{TSP}_0^*(\mathcal{S}_v)$ the length of the shortest Hamiltonian path over $\mathcal{S}_v$, which, under the triangle inequality, equals the optimal solution of a Traveling Salesman Problem with an artificial zero-distance depot.

Lemma 3. *The Maximum Spanning Star approximation is an upper bound on the length of the shortest Hamiltonian path:*

$$\text{MAXSPS}(\mathcal{S}_v) \geq \text{TSP}_0^*(\mathcal{S}_v).$$

Proof of Lemma 3. Denote by $C(\mathcal{S}_v) = \sum_{i,j \in \mathcal{S}_v} d_{i,j}$ the cost of the directed clique on $\mathcal{S}_v$. Then:

$$\sum_{i \in \mathcal{S}_v} \text{SPS}_i(\mathcal{S}_v) = C(\mathcal{S}_v), \text{ which yields}$$
$$\text{MAXSPS}(\mathcal{S}_v) \geq \frac{C(\mathcal{S}_v)}{|\mathcal{S}_v|}.$$

Akiyama et al. (2004, p. 40) present the *Walecki decomposition* of a *complete undirected* graph K_n with odd $n \geq 3$ into $(n-1)/2$ Hamiltonian cycles. It follows that K_n with even $n \geq 2$ can be decomposed into $n/2$ Hamiltonian paths. This result gives us:

Case 1: $|\mathcal{S}_v| \geq 2$ is even. The complete *directed* graph on $\mathcal{S}_v$ can be decomposed into $|\mathcal{S}_v|$ directed Hamiltonian paths. Hence,

$$\text{TSP}_0^*(\mathcal{S}_v) \leq \frac{C(\mathcal{S}_v)}{|\mathcal{S}_v|}.$$

Case 2: $|\mathcal{S}_v| \geq 3$ is odd. The complete *directed* graph on $\mathcal{S}_v$ can be decomposed into $|\mathcal{S}_v| - 1$ directed Hamiltonian cycles. We can remove one of the $|\mathcal{S}_v|$ edges in any of these $|\mathcal{S}_v| - 1$ Hamiltonian cycles to obtain a Hamiltonian path. Thereby,

$$\text{TSP}_0^*(\mathcal{S}_v) \leq \frac{|\mathcal{S}_v| - 1}{|\mathcal{S}_v|} \frac{C(\mathcal{S}_v)}{|\mathcal{S}_v| - 1} = \frac{C(\mathcal{S}_v)}{|\mathcal{S}_v|}.$$

Thus, for both even and odd $|\mathcal{S}_v| \geq 2$ we have

$$\text{TSP}_0^*(\mathcal{S}_v) \leq \frac{C(\mathcal{S}_v)}{|\mathcal{S}_v|} \leq \text{MAXSPS}(\mathcal{S}_v). \qquad \square$$

We report on the approximation performance of the Maximum Spanning Star in Section 7.1. In particular, we show that the Maximum Spanning Star approximation is highly correlated with the actual routing distance for our data sets (correlation of more than 85%).

5.2. (Un)loading time approximation

Next, we focus on approximating the (un)loading times for each cluster. We assume until Section 5.3 that $d_{i,j} = 0$ for all $i, j \in \mathcal{S}$ for notational convenience. Given the service level requirements $s_i^{\min}$ and $s_i^{\max}$ for each station $i \in \mathcal{S}$ and starting inventories s_i^0, we define:

$$\begin{aligned} s_i^+ &= \max\left\{s_i^{\min} - s_i^0, 0\right\} && \forall i \in \mathcal{S} \\ s_i^- &= \max\left\{s_i^0 - s_i^{\max}, 0\right\} && \forall i \in \mathcal{S} \end{aligned}$$

representing the minimum required number of bike deliveries (s_i^+) or bike pick ups (s_i^-) for each station to meet its service level requirements. An optimal vehicle route may require more bike deliveries and pickups than strictly necessary to decrease distance. Hence, using these minimum required bike deliveries and pickups, we can derive a lower bound on the (un)loading times component of $H^*(\mathcal{S}_v, \{v\})$.

Base case. (Un)loading times are in any case bounded from below by the minimum required number of bike deliveries and pickups.

$$H^*(\mathcal{S}_v, \{v\}) \geq \sum_{i \in \mathcal{S}_v} d^+ s_i^+ + d^- s_i^-$$

Case 1: $\sum_{i \in \mathcal{S}_v} \left(s_i^+ - s_i^-\right) - q_v^0 \geq 0$. In this case more bike deliveries than bike pickups plus vehicle starting inventory q_v^0 are required. Hence, additional bikes need to be picked up to accommodate all deliveries.

$$H^*(\mathcal{S}_v, \{v\}) \geq \sum_{i \in \mathcal{S}_v} \left(d^+ s_i^+ + d^- s_i^+\right) - d^- q_v^0$$

Case 2: $\sum_{i\in\mathcal{S}_v}\left(s_i^+ - s_i^-\right) - (Q_v - q_v^0) \leq 0$**.** In this case more bike pickups than bike deliveries plus vehicle slack capacity $(Q_v - q_v^0)$ are required. Hence, additional bikes need to be delivered to free up capacity for all pickups.

$$H^*(\mathcal{S}_v, \{v\}) \geq \sum_{i\in\mathcal{S}_v}\left(d^+ s_i^- + d^- s_i^-\right) - d^+\left(Q_v - q_v^0\right)$$

Note that when used in conjunction with the base case lower bound, the lower bounds for case 1 and 2 are non-binding when their conditions are not applicable.

5.3. Clustering Problem

We now implement the Maximum Spanning Star and (un)loading time approximations in our Clustering Problem. The objective is to minimize the estimated makespan $\hat{H}$ such that service level requirements can be satisfied using only within-cluster vehicle routing.

$$\text{minimize } \hat{H} \tag{P2}$$

s.t.

$$\sum_{v\in\mathcal{V}} z_{i,v} = 1 \qquad \forall i \in \mathcal{S}\setminus\mathcal{S}_0 \tag{14}$$

$$\sum_{v\in\mathcal{V}} z_{i,v} \leq 1 \qquad \forall i \in \mathcal{S}_0 \tag{15}$$

$$q_v^0 + \sum_{i\in\mathcal{S}} s_i^0 z_{i,v} \geq \sum_{i\in\mathcal{S}} s_i^{\min} z_{i,v} \qquad \forall v \in \mathcal{V} \tag{16}$$

$$-(Q_v - q_v^0) + \sum_{i\in\mathcal{S}} s_i^0 z_{i,v} \leq \sum_{i\in\mathcal{S}} s_i^{\max} z_{i,v} \qquad \forall v \in \mathcal{V} \tag{17}$$

$$\hat{h}_v \geq \sum_{j\in\mathcal{S}} d_{i,j}(z_{i,v} + z_{j,v} - 1) + \sum_{j\in\mathcal{S}}\left(d^+ s_j^+ + d^- s_j^-\right) z_{j,v} \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \tag{18}$$

$$\hat{h}_v \geq \sum_{j\in\mathcal{S}} d_{i,j}(z_{i,v} + z_{j,v} - 1) + \sum_{j\in\mathcal{S}}\left(d^+ s_j^+ + d^- s_j^+\right) z_{j,v} \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \tag{19}$$

$$\begin{aligned}
& - d^- q_v^0 & & \\
\hat{h}_v &\geq \sum_{j \in \mathcal{S}} d_{i,j}(z_{i,v} + z_{j,v} - 1) & \forall i \in \mathcal{S}, v \in \mathcal{V} & \quad (20) \\
& + \sum_{j \in \mathcal{S}} \left(d^+ s_j^- + d^- s_j^- \right) z_{j,v} & & \\
& - d^+ \left(Q_v - q_v^0 \right) & & \\
\hat{H} &\geq \hat{h}_v & \forall v \in \mathcal{V} & \quad (21) \\
z_{i,v} &\in \{0, 1\} & \forall i \in \mathcal{S}, v \in \mathcal{V} & \\
\hat{H}, \hat{h}_v &\geq 0 & \forall v \in \mathcal{V} &
\end{aligned}$$

Constraints (14)–(17) ensure feasibility of the clustering solution. Insufficient stations *must* be visited by a vehicle (14) and self-sufficient stations *can* be visited (15). A vehicle cluster must contain enough bikes, possibly using the vehicle starting inventory q_v^0, such that service level requirements can be met through within-cluster repositioning (16). Constraints (17) are similar but then for the maximum inventory in the cluster, possibly using vehicle surplus capacity $Q_v - q_v^0$.

Constraints (18) impose $\text{SPS}_i(\mathcal{S}_v)$ plus the base case (un)loading times bound as a lower bound on the estimated routing costs $\hat{h}_v$, if station $i \in \mathcal{S}$ is assigned to vehicle $v \in \mathcal{V}$. Since $\hat{h}_v$ is indirectly minimized, we have:

$$\hat{h}_v \geq \text{MAXSPS}(\mathcal{S}_v) + \sum_{i \in \mathcal{S}_v} d^+ s_i^+ + d^- s_i^-$$

Constraints (19) and (20) do the same but for case 1 and 2 from Section 5.2, respectively. The estimated makespan $\hat{H} = \max_{v \in \mathcal{V}} \hat{h}_v$ is linearized through constraints (21).

Note that we have formulated a compact clustering model with routing costs approximation, which guarantees that the service level requirements can be satisfied at each station while using only within-cluster vehicle routing.

Example 2. Figure 3 shows the assignment of stations to the two Hubway vehicles obtained by solving the Clustering Problem (P2) for the service level requirements from Example 1. We use real inventory data, based on a snapshot of the system taken at 8AM on Friday June 1st 2012.

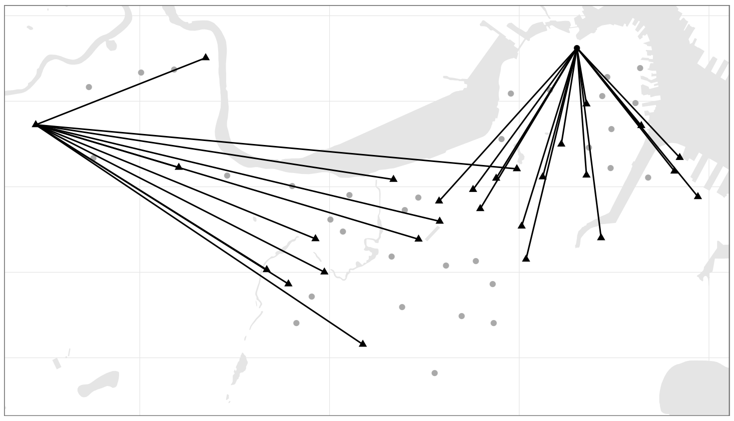

Figure 3: Example of a solution of the Clustering Problem (P2) for Hubway (Boston, MA).

5.4. Heuristics

Having presented MIP formulations for the Clustering Problem and the Routing Problem, we now formally introduce our Clustered Routing heuristics.

Heuristic 1 (Clustered MIP)**.** Subsequently:

1. Solve the Clustering Problem (P2).
2. For each $v \in \mathcal{V}$ solve the Routing Problem (P1) with $\mathcal{S} = \mathcal{S}_v$ and $\mathcal{V} = \{v\}$ to obtain $H^*(\mathcal{S}_v, \{v\})$.
3. $\tilde{H} = \max_{v\in\mathcal{V}} H^*(\mathcal{S}_v, \{v\})$.

Since the routing costs approximation is imperfect and may therefore lead to sub-optimal clusters, the Clustered MIP heuristic has an optimality loss. However, we can use the information obtained from routing the individual clusters to add *cuts* to the Clustering Problem.

Assume that an unknown optimal solution exists with makespan H^{OPT} strictly less than our best found solution $\tilde{H}$. This implies that all clusters $\mathcal{S}_v$ with known optimal routing costs $H^*(\mathcal{S}_v, \{v\}) \geq \tilde{H}$ are *not* part of the optimal solution. Therefore, these clusters can be eliminated from the solution space of the Clustering Problem. Note that, contrary to problems in which total routing costs are minimized, these cuts do not need to be removed later.

Heuristic 2 (Clustered MIP with Cuts)**.** Subsequently:

1. Define a cut set $\mathcal{C} \subseteq 2^{\mathcal{S}}$ of subsets of $\mathcal{S}$. Initialize $\mathcal{C} = \emptyset$.
2. Solve the Clustering Problem (P2) with additional constraints:

$$\sum_{i \in c} z_{i,v} - \sum_{i \in \mathcal{S} \setminus c} z_{i,v} \leq |c| - 1 \qquad \forall c \in \mathcal{C}, v \in \mathcal{V} \tag{22}$$

3. For each $v \in \mathcal{V}$ solve the Routing Problem (P1) with $\mathcal{S} = \mathcal{S}_v$ and $\mathcal{V} = \{v\}$ to obtain $H^*(\mathcal{S}_v, \{v\})$.
   (a) If $\max_{v \in \mathcal{V}} H^*(\mathcal{S}_v, \{v\}) < \tilde{H}$ or $\mathcal{C} = \emptyset$ then store the routing solution and (re)define $\tilde{H} = \max_{v \in \mathcal{V}} H^*(\mathcal{S}_v, \{v\})$.
   (b) For each $v \in \mathcal{V}$ with $H^*(\mathcal{S}_v, \{v\}) \geq \tilde{H}$ redefine $\mathcal{C} = \mathcal{C} \cup \{\mathcal{S}_v\}$.
4. If cuts were added in 3(b) go to Step 2.

Note that $\tilde{H}$ is only redefined if an improvement is found 3(a), in which case at least one cut is added per iteration (we have $H^*(\mathcal{S}_v, v) = \tilde{H}$ for at least one $v \in \mathcal{V}$). If no improvement is found, then $\tilde{H}$ is not redefined and at least one cluster has actual costs strictly larger than the best found solution $\tilde{H}$. We add a cut for all non-improving clusters (3b), and continue. Note that any *strict* subset or superset of a cut $c \in \mathcal{C}$ is *not* eliminated by constraints (22). The cuts force the Clustering Problem to iteratively adjust the assignment of stations to vehicles, thereby mitigating the approximation error. After finitely, but possibly exponentially many steps, the heuristic identifies the optimal solution.

Unfortunately, to our knowledge no stronger terminating condition for Clustered MIP with Cuts heuristic than an exhaustive search of the Clustering Problem solution space exists. However, Table 1 in Section 7.2 shows how quickly improvements over the Clustered MIP heuristic are attained.

6. Constraint Programming

In this section we present our Constraint Programming (CP) model for the Bike Sharing Rebalancing Problem. Constraint Programming is among the state of the art for solving complex routing and scheduling problems, even though it applies a generic modeling and solving approach. In particular, CP has been applied before to constrained routing problems (Kilby & Shaw, 2006). Most industrial CP solvers combine constraint propagation with large neighborhood search for solving routing problems (Shaw, 1998).

In order to take advantage of the strengths of CP, it is common to represent routing problems as scheduling problems, by representing the visit of a location as an *activity*. An activity is a high-level CP modeling structure that implicitly defines integer variables for its start time, length, and end time, and a Boolean variable for its presence. Activities for which the presence is not fixed to true are called *optional activities*. Traveling the distance between two locations is represented by sequence-dependent setup times between the respective activities (for each pair of locations), i.e., if station j is visited directly after station i, we need to respect the distance $d_{i,j}$ as 'setup' time. In CP, activities impact *resources* which, in case of routing problems, correspond to the vehicles. For example, for each vehicle, we must ensure that no two activities overlap.

We next specify the details of our CP model, following the AIMMS notation for activities and resources (Roelofs & Bisschop, 2016). In particular, each activity `A` induces the variables `A.Start`, end time `A.End`, and presence `A.Present`, as explained above. For each station $i \in \mathcal{S}$ and vehicle $v \in \mathcal{V}$, we define optional activities `Pickup[i,v]` and `Delivery[i,v]`. Variables $y^-_{i,v}$ and $y^+_{i,v}$ represent the pickup, respectively delivery, amount for vehicle v at station i, as in our models above. The makespan objective is implemented by minimizing the maximum end time of all pickup and delivery activities. Our CP model then becomes:

$$\text{minimize } \max\left\{\max_{\substack{i\in\mathcal{S}\\ v\in\mathcal{V}}}\{\texttt{Pickup[i,v].End}\}, \max_{\substack{i\in\mathcal{S}\\ v\in\mathcal{V}}}\{\texttt{Delivery[i,v].End}\}\right\} \tag{P3}$$

s.t.

$$s_i^0 + \sum_{v\in\mathcal{V}} y^+_{i,v} - y^-_{i,v} \geq s_i^{\min} \qquad \forall i \in \mathcal{S} \tag{23}$$

$$s_i^0 + \sum_{v\in\mathcal{V}} y^+_{i,v} - y^-_{i,v} \leq s_i^{\max} \qquad \forall i \in \mathcal{S} \tag{24}$$

$$\sum_{v\in\mathcal{V}} \begin{matrix}\texttt{Pickup[i,v].Present}+\\ \texttt{Delivery[i,v].Present}\end{matrix} \leq 1 \qquad \forall i \in \mathcal{S} \tag{25}$$

$$\texttt{Pickup[i,v].Present} = 1 \Leftrightarrow y^-_{i,v} \geq 1 \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \tag{26}$$

$$\texttt{Pickup[i,v].Present} = 0 \Leftrightarrow y^-_{i,v} = 0 \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \tag{27}$$

$$\texttt{Delivery[i,v].Present} = 1 \Leftrightarrow y^+_{i,v} \geq 1 \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \tag{28}$$

$$\texttt{Delivery[i,v].Present} = 0 \Leftrightarrow y^+_{i,v} = 0 \qquad \forall i \in \mathcal{S}, v \in \mathcal{V} \qquad (29)$$
$$y^-_{i,v}, y^+_{i,v} \in \{0, \ldots, Q_v\} \qquad \forall i \in \mathcal{S}, v \in \mathcal{V}$$

with activities

```
Activity Pickup[i,v](
   Schedule domain:  $\{0, ..., \texttt{MaxTime}\}$
   Property:  Optional
   Length: $d^- y^-_{i,v}$
)
Activity Delivery[i,v](
   Schedule domain:  $\{0, ..., \texttt{MaxTime}\}$
   Property:  Optional
   Length: $d^+ y^+_{i,v}$
)
```

and resources

```
Sequential resource VehicleTime[v](
   Schedule domain: $\{0, ..., \texttt{MaxTime}\}$
   Activities:  Pickup[i,v], Delivery[i,v]
   Transition: $d_{i,j}$
)
Parallel resource VehicleInventory[v](
   Activities:  Pickup[i,v], Delivery[i,v]
   Level range: $\{0, \ldots, Q_v\}$
   Initial value: $q^0_v$
   Begin change:  Delivery[i,v]: $-y^+_{i,v}$
   End change:  Pickup[i,v]: $y^-_{i,v}$
)
Parallel resource StationInventory[i](
   Activities:  Pickup[i,v], Delivery[i,v]
   Level range: $\{0, \ldots, C_i\}$
   Initial value: $s^0_i$
   Begin change:  Delivery[i,v]: $y^+_{i,v}$
   End change:  Pickup[i,v]: $-y^-_{i,v}$
)
```

As before, constraints (23)–(24) impose the service level requirements from Lemma 1. Constraints (25) are so-called *alternative resource* constraints, which limit the number of visits to one per station. That is, the trucks fulfill the role of the alternative resources for visiting a station, and each station is allocated to at most one truck. Adding these constraints can greatly improve the performance of the constraint propagation, but the optimal solution may be eliminated. Note that it may not always be possible to impose the alternative resource constraints, for example due to limited vehicle capacity. In such cases, the model can trivially be extended to allow multiple visits by increasing the right-hand side. We can index the activities and variables $y^-_{i,v}, y^+_{i,v}$ correspondingly. However, our preliminary experimentation showed strongly decreasing computational performance if we allowed multiple visits per location. Therefore, we imposed the alternative resource constraints in our experiments. Constraints (26)–(29) link the vehicle presence constraints with performing a pickup or delivery. Note that the *if and only if* constraints enhance propagation.

For the optional activities `Pickup[i,v]` and `Delivery[i,v]` we set the length equal to the (un)loading times $d^- y^-_{i,v}$ and $d^+ y^+_{i,v}$.

For each vehicle we introduce two types of resources. The first represent the no-overlap conditions with respect to the vehicle time, using a `Sequential resource` named `VehicleTime[v]`. For each such resource, we identify the discrete time horizon as its `Schedule domain`, while the keyword `Activities` specifies which activities impact the resource. The arc-dependent transition times model the travel distances via `Transition`.

The second resource associated with a vehicle is its inventory, modeled as a `Parallel resource` named `VehicleInventory[v]`. In addition to specifying the set of activities in its scope, we define its `Level range` to be $\{0, \ldots, Q_v\}$, which is initialized at q^0_v. Furthermore, we specify for each activity in its scope how it impacts the level. For `Delivery[i,v]`, the level is changed at the start of the activity, with amount $-y^+_{i,v}$. Likewise, for `Pickup[i,v]`, the level is changed at the end of the activity with amount $y^-_{i,v}$.

Lastly, for each station we define a `Parallel resource` representing the station inventory, named `StationInventory[i]`. The range of this inventory is $\{0, \ldots, C_i\}$ with initial value s^0_i. Level changes for pickups and deliveries are exactly opposite to the vehicle inventory changes.

7. Computational Results

In this section we report on the performance of our routing costs approximation and heuristics. Recall that we give a detailed overview of our data sources in Appendix A.

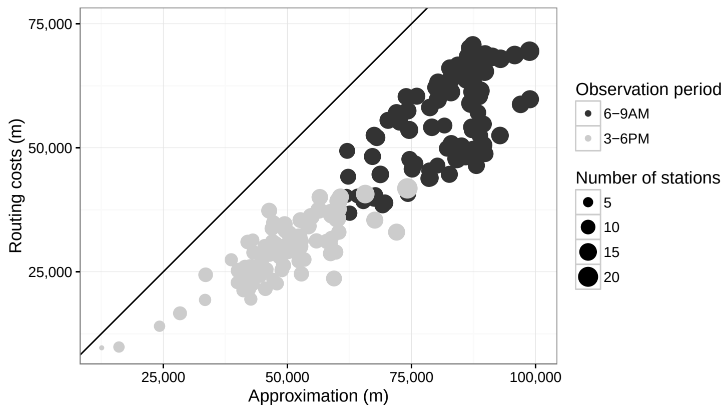

Figure 4: Computational results for the routing costs approximation.

Note. Routing costs $H^*(\mathcal{S}_v, \{v\})$ are plotted against the approximation $\hat{h}_v$ for 82 two-vehicle instances of Hubway (Boston, MA). Each data point corresponds to a cluster (i.e. 164 data points shown). The correlation of $H^*(\mathcal{S}_v, \{v\})$ and $\hat{h}_v$ equals 91.2%. The correlation between the *distance* $(d_{i,j})$ component of $H^*(\mathcal{S}_v, \{v\})$ and $\text{MAXSPS}(\mathcal{S}_v)$ is 87.4%.

7.1. Approximation performance

In Section 5.1 we proved that $\text{MAXSPS}(\mathcal{S}_v) \geq \text{TSP}_0^*(\mathcal{S}_v)$ to motivate our intuition that $\text{MAXSPS}(\mathcal{S}_v)$ approximates $H^*(\mathcal{S}_v, \{v\}) \geq \text{TSP}_0^*(\mathcal{S}_v)$ (assuming $d^- = d^+ = 0$). By adding an approximation for (un)loading costs given that $d^- > 0$ and $d^+ > 0$, we establish the routing costs approximation $\hat{h}_v$ in Clustering Problem (P2). Figure 4 visualizes the relationship between $H^*(\mathcal{S}_v, \{v\})$ and $\hat{h}_v$ with a scatterplot, in which equality is shown as reference line.

While we established that $H^*(\mathcal{S}_v, \{v\}) \geq \text{TSP}_0^*(\mathcal{S}_v)$, we note that the additional feasibility constraints on the vehicle route imposed in (P1) do *not* lead to routing costs higher than the approximation for any instance. Rather, the Maximum Spanning Star approximation consistently overestimates the routing costs. A linear regression on $H^*(\mathcal{S}_v, \{v\})$ with an imposed zero intercept estimates a .65 coefficient for $\hat{h}_v$ yielding $R^2 = 98.0\%$. Nonetheless, an approximation error is present. Next, we show how the improvement scheme with elimination cuts overcomes this limitation of the Clustered MIP heuristic.

7.2. Heuristics performance

We report computational results comparing the exact MIP (P1), Clustered MIP (Heuristic 1), Clustered MIP with Cuts (Heuristic 2), and CP (P3) approaches to the Bike Sharing Rebalancing Problem. We consider multiple families of instances based on real trip and inventory data provided by Hubway (Boston, MA) and Capital Bikeshare (Washington, DC).

All experiments were performed on an Intel Xeon X3323 @ 2.50GHz with 16GB of memory, using AIMMS 4.2 x64 modeling software with MIP solver GUROBI 6.0 (which outperformed CPLEX on test instances) and CP solver IBM ILOG CP Optimizer 12.6.

Parameters

A family of instances is defined by a market (Hubway or Capital Bikeshare), observation period (e.g. 6–9AM), service level (we consistently use 85% in line with practice), and number of vehicles. We observe that the morning and afternoon commute are the most challenging rebalancing problems, so use these as examples for our static rebalancing approach. Subsequently, for each family we generate multiple problem instances by using different *inventory snapshots* containing the starting inventory s_i^0 for each station $i \in \mathcal{S}$ at the beginning of the observation period.

We calculate Euclidian distances $d_{i,j} = d_{j,i}$ in meters based on the latitude and longitude of stations. We use $d^+ = d^- = 357$ meters, by converting the typical (un)loading time of 1 minute per bike to meters assuming an average speed of 30 km/h when driving and dividing by a detour factor of 1.4 (since our station distances are Euclidian, not road). We assume $q_v^0 = 0$ to refrain from random data generation. For MIP we set $|\mathcal{T}| = |\mathcal{S} \setminus \mathcal{S}_0|$ (this ensures feasibility for our instances) and for the Clustered MIP heuristics we set $|\mathcal{T}| = |\mathcal{S}_v| + 1$ (this allows a revisit).

We observed that the computation time for solving the exact MIP (P1) is greatly impacted by the variables $y^-_{i,t,v}$ and $y^+_{i,t,v}$ being integer. We therefore report here the results obtained when relaxing the integrality constraints on these variables. When the objective is to minimize the total distance, or when the (un)loading cost are ignored, relaxing the integrality does not impact the optimal solution. However, since our objective is to minimize makespan, the optimal solution may benefit from using fractional values for $y^-_{i,t,v}$ and $y^+_{i,t,v}$. We suggest a *rounding heuristic* to obtain a valid upper bound: start with the vehicle with the shortest route and one or more fractional pickups/deliveries, round up while maintaining vehicle inventory feasibility, continue with next vehicle until solution is integer.

Hubway (Boston, MA)

We restrict to $|\mathcal{S}| = 60$ stations to obtain sufficient trip observations to calibrate the service level requirements on 82 weekdays between November 1st 2011 and May 31st 2012. We use 41 inventory snapshots on weekdays between June 1st and July 29th 2012. Hubway currently operates $|\mathcal{V}| = 3$ vehicles with capacity $Q_v = 22$ since opening an additional 40 stations in Summer 2012. However, during our observations Hubway only operated two vehicles. We investigate both truck fleet sizes to reveal the possible implications for performance.

After extensive experiments we set the computational cut-offs (if unsolved): MIP after 7200 seconds; Clustered MIP after 20 seconds for (P2) and 120 seconds for (P1); Clustered MIP with Cuts after 300 seconds in total with 20 seconds for (P2) and 120 seconds for (P1); CP after 300 seconds. For CP we set the schedule domain with `MaxTime` = 250000, which proved necessary to quickly identify a feasible (but low-quality) solution.

Table 1 summarizes our computational results for Hubway. We show the average solutions and computation times per instance family for the exact MIP and our heuristics. For our improvement scheme (Clustered MIP with Cuts), we also show the average number of iterations completed.

We observe that in 5 minutes our Clustered MIP with Cuts and CP heuristics outperform the best found solution of the MIP after two hours with 25–45% on average for the two-vehicle families. For the three-vehicle families, the average improvement is 50–60%. In addition, we note that our heuristics found feasible solutions in 5 minutes for 27 instances (out of 164) for which the solver did not yield an integer MIP solution after two hours (see Table 2).

We can see that for the Clustered MIP heuristic, computational complexity is still in the routing problem, since solution times are shorter (and number of iterations higher) for the three-vehicle instance families.

In Figure 5 we present a more detailed comparison of our dedicated Clustered MIP with Cuts heuristic with the CP heuristic for the Hubway instance families. We show how they perform in comparison to the best found MIP solution after two hours (depicted as 100%). We note that the performance of the full MIP model decreases strongly for the three vehicle families, with our 5 minute heuristic solutions up to 75% better than the best found MIP

Table 1: Results for Hubway (Boston, MA).

				MIP			*Clustered MIP*		*Clustered MIP with Cuts*			*CP*	
Family	$\lvert\mathcal{V}\rvert$	Inst.	$\lvert\mathcal{S} \setminus \mathcal{S}_0\rvert$	LP bound	Best found	Time	Solution	Time	Solution	Iterations	Time	Solution	Time
6–9AM	2	32	29	49902	114431	7204	61545	137	60854	3	300	63145	300
6–9AM	3	41	29	32971	96363	7207	40190	44	38948	10	300	43985	300
3–6PM	2	27	21	24888	41290	7203	30829	32	29533	38	300	32069	300
3–6PM	3	37	22	16824	46120	7205	20956	10	20043	34	300	23894	300

Note. These results are averaged over the instances of each instance family for which the solver was able to find an integer MIP solution in two hours. The $\lvert\mathcal{S} \setminus \mathcal{S}_0\rvert$ column shows the average number of insufficient stations of the instance family. For results of the 27 instances without MIP solution, see Table 2.

Table 2: Results for Hubway (Boston, MA) instances without a MIP solution.

				MIP		*Clustered MIP*		*Clustered MIP with Cuts*			*CP*	
Family	$\lvert\mathcal{V}\rvert$	Inst.	$\lvert\mathcal{S} \setminus \mathcal{S}_0\rvert$	LP bound	Time	Solution	Time	Solution	Iterations	Time	Solution	Time
6-9AM	2	9	29	50604	7203	62844	140	62473	2	300	66649	300
3-6PM	2	14	24	27746	7202	33825	62	33148	16	300	35453	300
3-6PM	3	4	24	18094	7204	23453	10	21954	28	300	25079	300

Note. These results are averaged over the instances of each instance family for which the solver was unable to find an integer MIP solution.

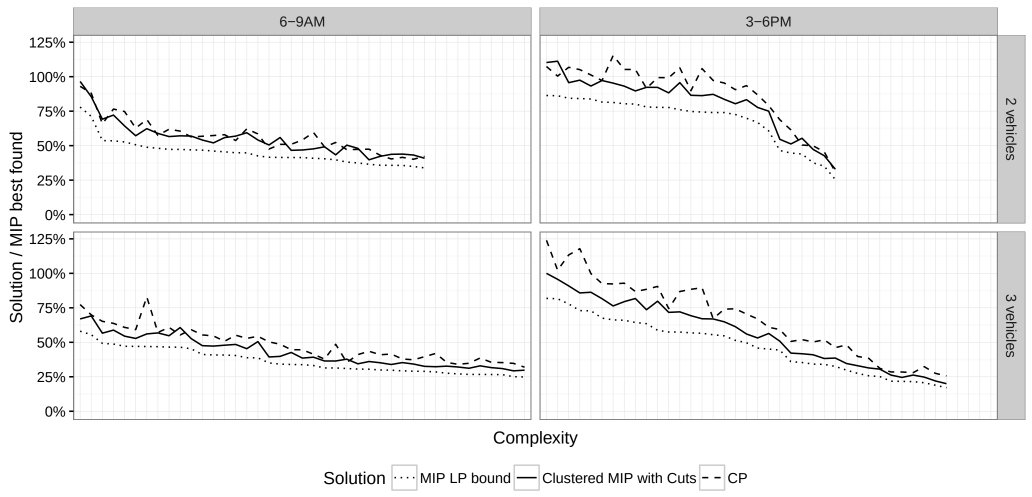

Figure 5: Computational results for Hubway (Boston, MA).

Note. Instances are sorted with decreasing MIP LP bound to MIP best found solution ratio, to approximate complexity. Here, the MIP best found solution is shown as 100%. Instances without MIP solution not shown (see Table 2).

solutions after two hours.

CP performs very well for the Hubway instances, with solutions very close to those of the Clustered MIP with Cuts heuristic. We believe that CP is able to quickly identify a feasible solution which is subsequently improved. CP performance appears better for the 6–9AM instances, which could be driven by the higher number of insufficient stations, making the problem more focused on scheduling.

Capital Bikeshare (Washington, DC)

We restrict to $|\mathcal{S}| = 135$ stations to obtain 130 trip observations on weekdays between January 1st and June 30th 2012. We use 30 8AM and 28 4PM inventory snapshots of weekdays obtained between December 17th 2012 and January 30th 2013. Capital Bikeshare operated $|\mathcal{V}| = 5$ vehicles with $Q_v = 25$ during the observation period.

With 135 stations and 5 vehicles, the Capital Bikeshare instances were too complex to derive feasible solutions or even useful LP bounds from the full MIP (P1) within a reasonable amount of time. Therefore, we report only on the performance of our heuristics.

Table 3: Results for Capital (Washington, DC).

				Clustered MIP		*Clustered MIP with Cuts*			*CP*	
Family	$\lvert\mathcal{V}\rvert$	Inst.	$\lvert\mathcal{S}\setminus\mathcal{S}_0\rvert$	Solution	Time	Solution	Iterations	Time	Solution	Time
6-8AM	5	22	27	28722	21	26914	14	300	47512	300
3-5PM	5	28	29	23812	23	21772	13	300	35675	300

Note. These results are averaged over the instances of each instance family for which the solver was able to find a CP solution. The $|\mathcal{S}\setminus\mathcal{S}_0|$ column shows the average number of insufficient stations of the instance family. For results of the 8 instances without CP solution, see Table 4.

Table 4: Results for Capital (Washington, DC) instances without a CP solution.

				Clustered MIP		*Clustered MIP with Cuts*		
Family	$\lvert\mathcal{V}\rvert$	Inst.	$\lvert\mathcal{S}\setminus\mathcal{S}_0\rvert$	Solution	Time	Solution	Iterations	Time
6-8AM	5	8	30	34939	21	32550	13	300

Note. These results are averaged over the instances of each instance family for which the solver was unable to find a feasible CP solution in 5 minutes.

Table 3 summarizes our computational results for Capital. On average the Clustered MIP with Cuts heuristic performs 40–45% better than CP. This highlights where we believe our cluster-first route-second heuristic excels.

The polynomial-size Clustering Problem (P2) allows rapid decomposition of the multi-vehicle problem into reasonably good single-vehicle clusters. Then, the single-vehicle Routing Problem (P1) can be quickly solved to optimality for each cluster. The subsequent improvement cuts mitigate both the approximation error (see Figure 4) and potential sub-optimality incurred from cutting off the Clustering Problem before the solver is finished.

In addition, we note that our Clustered MIP heuristics found solutions for 8 Capital instances (out of 58) for which the solver did not yield a feasible constraint programming solution after 5 minutes (see Table 4). This is potentially because of the alternative resource constraint, which prevents revisits.

In Figure 6 we present a more detailed comparison of our heuristics, with the Clustered MIP heuristic shown as 100%. For some instances, even the simple Clustered MIP heuristic is up to 50% better than CP. Figure 6 also shows clearly how adding cuts in the Clustered MIP with Cuts heuristic can

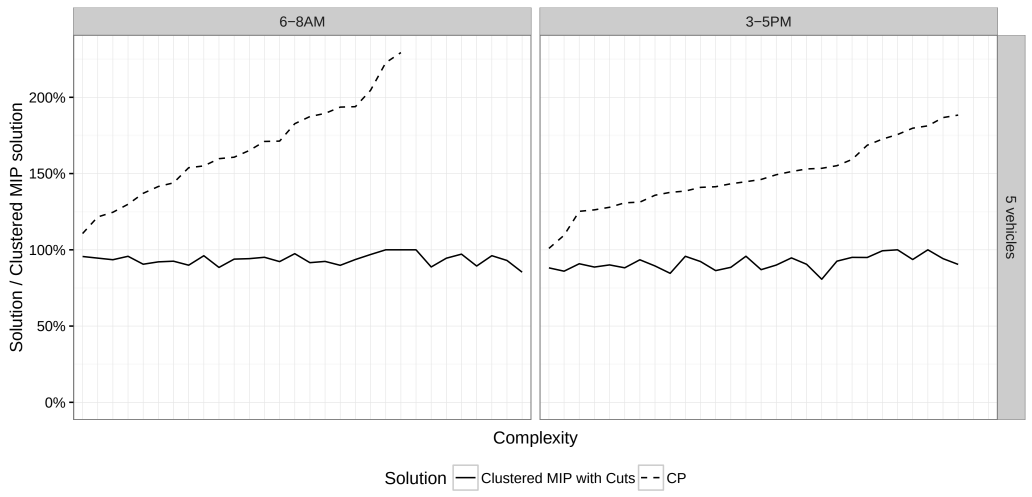

Figure 6: Computational results for Capital Bikeshare (Washington, DC).

Note. Instances are sorted with increasing CP solution to Clustered MIP solution ratio, to approximate complexity. The Clustered MIP solution is shown as 100%.

yield improvements of up to 15% over the simple Clustered MIP heuristic.

The results for Capital Bikeshare show that our dedicated Clustered MIP (with Cuts) heuristic performs better than CP, especially for instances with a larger vehicle fleet and a lower number of stations per vehicle. This implies the polynomial-size Clustering Problem can handle large sets of insufficient stations, when sufficient vehicles are available to divide the workload. When an instance is more similar to a scheduling problem (i.e. lower number of vehicles and longer routes, like for some Hubway instances), the techniques embedded in CP show their strength. Combining a MIP clustering heuristic with constraint programming for the single-vehicle routing (i.e. scheduling) problems could be an interesting area for future research.

8. Conclusions

This paper is the first to unify dual-bounded service level constraints (which add inventory flexibility) and vehicle routing for static rebalancing in bike sharing systems. We represent the inventory at each station as a finite-buffer single-server non-stationary queuing system and use the Kolmogorov forward equations to calculate service level requirements. We introduce the notion of self-sufficient stations, which fulfill these requirements with their starting inventory. Hence, self-sufficient stations do not necessarily need to be visited by a vehicle, but may act as source or sink nodes.

We present a mixed-integer programming based Clustering Problem that decomposes the multi-vehicle rebalancing problem into single-vehicle problems, while taking into account service level feasibility constraints (a cluster-first route-second approach). We introduce a novel polynomial-size Maximum Spanning Star routing costs approximation for the Clustering Problem to achieve fast computational performance. We develop an improvement scheme based on elimination cuts to mitigate the approximation error. Furthermore, we provide the first constraint programming formulation of the bike sharing rebalancing problem.

Using empirical data from two bike sharing systems, we extensively test the Clustered MIP heuristics against the classical full MIP model and the constraint programming approach. Our Clustered MIP with Cuts heuristic outperforms the constraint programming formulation as the number of vehicles increases. Constraint programming performs well when the number of vehicles is low and the number of stations per vehicle is high, which makes the problem more focused on scheduling. Both the Clustered MIP and constraint programming approaches identify better solutions within 5 minutes than the often-used full MIP after two hours. We thus believe that our approach is suitable for practical implementation in bike sharing systems.

The novel heuristics and our approximation may also be applicable to other constrained routing problems, specifically to (extended) One-Commodity Pickup-and-Delivery VRPs like the empty freight container rebalancing problem, as well as to other sharing systems.

Appendix A. Data sources

In order to produce the examples and computational results, we processed two data sets from Hubway (Boston, MA) and Capital Bikeshare (Washington, DC), which have identical formatting. These data sets were made available through their websites `thehubway.com` and `capitalbikeshare.com`. Each data set consists of three tables: `Stations`, `Trips` and `Snapshots`.

The `Stations` table contains the following fields for each station: `id`, `name`, `lat`, `lng`, `installed`, `locked` and `temporary`. We use the `id` field to create the relationship with the `Trips` and `Inventory` tables. We use the `lat` and `lng` fields to calculate the Euclidean distance matrix d in meters with the `spDists` function of the `sp` package in R.

The `Trips` table has the following fields for each trip: `id`, `start_date`, `end_date`, `start_station`, `end_station`, `bike_name`, `bike` and `member_type`. We calculate the number of pickups at a station during the observation period using `start_date` and `start_station`, and the number of returns using `end_date` and `end_station`, to synchronize these events. As mentioned in Section 7.2, we select a subset of the stations for calculating the service level requirements, to ensure that sufficient trip observations are available.

The `Snapshots` table contains the following fields for each station/date combination: `id`, `bikes`, `docks` and `date`. We calculate `capacity` = `bikes`+ `docks` for each snapshot, since reparations or extensions may lead to changes in the station capacity over time. We set C_i equal to the maximum `capacity` of station $i \in \mathcal{S}$ to prevent infeasibilities. We web scraped the `Snapshots` table for Capital Bikeshare from
`http://capitalbikeshare.com/data/stations/bikeStations.xml`.

Acknowledgements

The authors would like to thank Scott Mullen, general manager of Hubway, and Eric Gilliland, general manager of Capital Bikeshare, for their guidance and information on bike sharing operations.

References

Akiyama, J., Kobayashi, M., & Nakamura, G. (2004). Symmetric Hamilton cycle decompositions of the complete graph. *Journal of Combinatorial Designs*, *12*, 39–45. doi:`10.1002/jcd.10066`.

Benchimol, M., Benchimol, P., Chappert, B., de la Taille, A., Laroche, F., Meunier, F., & Robinet, L. (2011). Balancing the stations of a self service ‘bike hire’ system. *RAIRO - Operations Research*, *45*, 37–61. doi:`10.1051/ro/2011102`.

Borgnat, P., & Abry, P. (2009). Studying Lyon’s Vélo’V: A Statistical Cyclic Model. In *European Conference on Complex Systems* (pp. 1–6).

Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-B., & Fleury, E. (2011). Shared Bicycles in a City: a Signal Processing and Data Analysis

Perspective. *Advances in Complex Systems*, *14*, 415–438. doi:10.1142/S0219525911002950.

Charikar, M., Goemans, M. X., & Karloff, H. (2004). On the Integrality Ratio for Asymmetric TSP. In *45th Annual IEEE Symposium on Foundations of Computer Science* (pp. 101–107). IEEE. doi:10.1109/FOCS.2004.45.

Chemla, D., Meunier, F., & Calvo, R. W. (2013). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, *10*, 120 – 146. doi:10.1016/j.disopt.2012.11.005.

Contardo, C., Morency, C., & Rousseau, L.-M. (2012). *Balancing a Dynamic Public Bike-Sharing System.* Technical Report CIRRELT.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Operations Research*, *2*, 393–410. doi:10.1287/opre.2.4.393.

Dell'Olio, L., Ibeas, A., & Moura, J. L. (2011). Implementing bike-sharing systems. *Proceedings of the ICE - Municipal Engineer*, *164*, 89–101. doi:10.1680/muen.2011.164.2.89.

DeMaio, P. (2009). Bike-sharing: History, Impacts, Models of Provision, and Future. *Journal of Public Transportation*, *12*, 41–56.

Di Gaspero, L., Rendl, A., & Urli, T. (2013a). A Hybrid ACO+CP for Balancing Bicycle Sharing Systems. In M. J. Blesa, C. Blum, P. Festa, A. Roli, & M. Sampels (Eds.), *Hybrid Metaheuristics: 8th International Workshop* (pp. 198–212). doi:10.1007/978-3-642-38516-2_16.

Di Gaspero, L., Rendl, A., & Urli, T. (2013b). Constraint-Based Approaches for Balancing Bike Sharing Systems. In C. Schulte (Ed.), *Proceedings of CP* (pp. 758–773). volume 8124 of *Lecture Notes in Computer Science*. doi:10.1007/978-3-642-40627-0_56.

Di Gaspero, L., Rendl, A., & Urli, T. (2015). Balancing bike sharing systems with constraint programming. *Constraints*, *21*, 1–31. doi:10.1007/s10601-015-9182-1.

Erdoğan, G., Battarra, M., & Calvo, R. W. (2015). An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, *245*, 667–679.

Erdoğan, G., Laporte, G., & Calvo, R. W. (2013). *The One-Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals*. Technical Report CIRRELT.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*, 109–124. doi:`10.1002/net.3230110205`.

Forma, I. A., Raviv, T., & Tzur, M. (2015). A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation research part B: methodological*, *71*, 230–247.

Fricker, C., & Gast, N. (2014). Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, (pp. 1–31). doi:`10.1007/s13676-014-0053-5`.

Fricker, C., Gast, N., & Mohamed, H. (2012). Mean field analysis for inhomogeneous bike sharing systems. In *DMTCS Proceedings* (pp. 365–376).

Froehlich, J., & Oliver, N. (2008). Measuring the pulse of the city through shared bicycle programs. In *Proceedings of International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems* (pp. 16–20).

George, D. K., & Xia, C. H. (2011). Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research*, *211*, 198–207.

Hampshire, R. C., Marla, L., & Eluru, N. (2013). *An Empirical Analysis of Bike Sharing Usage and Rebalancing: Explaining Trip Generation and Attraction from Revealed Preference Data*. Technical Report Heinz College, Carnegie Mellon University.

Held, M., & Karp, R. M. (1970). The Traveling-Salesman Problem and Minimum Spanning Trees. *Oper. Res.*, *18*, 1138–1162. doi:`10.1287/opre.18.6.1138`.

Hernández-Pérez, H., & Salazar-González, J.-J. (2003). The One-Commodity Pickup-and-Delivery Travelling Salesman Problem. In M. Jünger, G. Reinelt, & G. Rinaldi (Eds.), *Combinatorial Optimization – Eureka,* *You Shrink!* (pp. 89–104). Springer Berlin Heidelberg volume 2570 of *Lecture Notes in Computer Science*. doi:10.1007/3-540-36478-1_10.

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J., & Banchs, R. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, *6*, 455–466. doi:10.1016/j.pmcj.2010.07.002.

Kendall, D. G. (1953). Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, *24*, 338–354. doi:10.1214/aoms/1177728975.

Kilby, P., & Shaw, P. (2006). Vehicle Routing. In F. Rossi, P. Van Beek, & T. Walsh (Eds.), *Handbook of Constraint Programming* chapter 23. (pp. 801–836). Elsevier.

Kloimüllner, C., Papazek, P., Hu, B., & Raidl, G. R. (2015). A Cluster-First Route-Second Approach for Balancing Bicycle Sharing Systems. In *Computer Aided Systems Theory–EUROCAST 2015* (pp. 439–446). doi:10.1007/978-3-319-27340-2_55.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*, 231–247. doi:10.1016/0377-2217(92)90138-Y.

Laporte, G., Meunier, F., & Calvo, R. W. (2015). Shared mobility systems. *4OR*, *13*, 341–360.

Lathia, N., Ahmed, S., & Capra, L. (2012). Measuring the impact of opening the London shared bicycle scheme to casual users. *Transportation Res. C*, *22*, 88–102. doi:10.1016/j.trc.2011.12.004.

Leurent, F. (2012). Modelling a vehicle-sharing station as a dual waiting system: stochastic framework and stationary analysis. Working paper.

Lin, J.-R., & Yang, T.-H. (2011). Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review*, *47*, 284–294. doi:10.1016/j.tre.2010.09.004.

Martinez, L. M., Caetano, L., Eiró, T., & 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*, 513–524. doi:`10.1016/j.sbspro.2012.09.769`.

Meddin, R., & DeMaio, P. (2016). The Bike Sharing World Map. URL: `http://www.metrobike.net/`.

Morse, P. M. (1958). *Queues, Inventories, And Maintenance: The Analysis Of Operational Systems With Variable Demand And Supply*. (Dover ed.). New York: Wiley.

Nair, R., & Miller-Hooks, E. (2011). Fleet Management for Vehicle Sharing Operations. *Transportation Sci.*, *45*, 524–540. doi:`10.1287/trsc.1100.0347`.

Nair, R., Miller-Hooks, E., Hampshire, R. C., & Bušić, A. (2013). Large-Scale Vehicle Sharing Systems: Analysis of Vélib'. *International Journal of Sustainable Transportation*, *7*, 85–106. doi:`10.1080/15568318.2012.660115`.

New York City Bike Share (2013). Frequently Asked Questions. URL: `http://a841-tfpweb.nyc.gov/bikeshare/faq/#who-is-running-bike-share-in-nyc`.

O'Mahony, E. D. (2015). *Smarter Tools For (Citi) Bike Sharing*. Ph.D. thesis Cornell University.

Prem Kumar, V., & Bierlaire, M. (2012). Optimizing Locations for a Vehicle Sharing System. In *Swiss Transport Research Conference* (pp. 1–30).

Rainer-Harbach, M., Papazek, P., Raidl, G. R., Hu, B., & Kloimüllner, C. (2014). PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, *63*, 597–629. doi:`10.1007/s10898-014-0147-5`.

Raviv, T., & Kolka, O. (2013). Optimal inventory management of a bike-sharing station. *IIE Transactions*, *45*, 1077–1093. doi:`10.1080/0740817X.2013.770186`.

Raviv, T., Tzur, M., & Forma, I. A. (2013). Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, *2*, 187–229. doi:10.1007/s13676-012-0017-6.

Roelofs, M., & Bisschop, J. (2016). *AIMMS – The Language Reference*. AIMMS.

Shaheen, S., Guzman, S., & Zhang, H. (2010). Bikesharing in Europe, the Americas, and Asia. *Transportation Research Record: Journal of the Transportation Research Board*, *2143*, 159–167. doi:10.3141/2143-20.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher, & J.-F. Puget (Eds.), *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming* (pp. 417–431). Springer-Verlag.

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

Vogel, P., Greiser, T., & Mattfeld, D. C. (2011). Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns. *Procedia - Social and Behavioral Sciences*, *20*, 514–523. doi:10.1016/j.sbspro.2011.08.058.

Vogel, P., & Mattfeld, D. C. (2010). Modeling of repositioning activities in bike-sharing systems. *World Conference on Transport Research*, (pp. 1–13).

Waserhole, A., & Jost, V. (2012). Vehicle Sharing System Pricing Regulation: Transit Optimization of Intractable Queuing Network. Working paper.

Wu, B. Y., Lancia, G., Bafna, V., Chao, K.-M., Ravi, R., & Tang, C. Y. (1998). A polynomial time approximation scheme for minimum routing cost spanning trees. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms* (pp. 21–32). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.