Technical Report AC-TR-15-004

November 2015

# Full-Load Route Planning for Balancing Bike Sharing Systems by Logic-Based Benders Decomposition

Christian Kloimüllner and Günther R. Raidl

This is the author's copy of a paper that has been submitted to the journal Networks for a special issue dedicated to the VeRoLog conference which took place from June 8–10, 2015 in Vienna.

*www.ac.tuwien.ac.at/tr*

Technical Report AC-TR-15-004

# Full-Load Route Planning for Balancing Bike Sharing Systems by Logic-Based Benders Decomposition

Christian Kloimüllner
Günther R. Raidl

Institute of Computer Graphics and Algorithms, TU Wien

Favoritenstraße 9–11/1861, 1040 Vienna, Austria

{kloimuellner|raidl}@ac.tuwien.ac.at

November 30, 2016

Public Bike Sharing Systems require some kind of rebalancing to avoid too many rental stations of running empty or entirely full, which would make the system ineffective and annoy customers. Most frequently, a fleet of vehicles with trailers is used for this purpose, moving bikes among the stations. Previous works considered different objectives and modeled the underlying routing problem in different ways, but they all allow an arbitrary number of bikes to be picked up at some stations and delivered to other stations, just limited by the vehicles' capacities. Observations in practice, however, indicate that in larger well-working bike sharing systems drivers almost never pickup or deliver only few bikes, but essentially always approximately full vehicle loads. Many stations even require several visits with full loads. Due to budgetary reasons, typically only just enough drivers and vehicles are employed to achieve a reasonable balance most of the time, but basically never an ideal one where single bikes play a substantial role. Consequently, we investigate here a simplified problem model, in which only full vehicle loads are considered for movement among the rental stations. This restriction appears to have only a minor impact on the achieved quality of the rebalancing in practice but eases the modeling substantially. More specifically, we formulate the rebalancing problem as a *selective unit-capacity pickup and delivery problem with time budgets on a bipartite graph* and present a compact mixed integer linear programming model, a logic-based Benders decomposition and a variant thereof, namely

Branch-and-Check for it. For the general case, instances with up to 70 stations, and for the single-vehicle case instances with up to 120 stations are solved to proven optimality. A comparison to leading metaheuristic approaches considering flexible vehicle loads indicates that indeed the restriction to full loads has only a very small impact on the finally achieved balance in typical scenarios of Citybike Wien.

***Keywords:*** balancing bike-sharing systems, logic-based Benders decomposition, Branch-and-Check, vehicle routing, pickup-delivery, traveling salesman problem

Technical Report AC-TR-15-004

## 1. Introduction

Public bike sharing systems (PBSs) provide a modern way of shared public transport within cities. These systems consist of *rental stations* distributed in parts of a city. In state-of-the-art PBSs every station has a self-service computer terminal authenticating the customers, and ideally also used to allow instant registration for new clients. Customers have to authenticate and provide a payment method to reduce theft and vandalism. Rental stations consist of *slots* which can either be empty or occupied by a bike. These slots are connected to the whole computer system allowing the operators as well as the customers to have an overview of the status of each station. If there is at least one slot occupied by a bike, customers have the opportunity to rent a bike via the terminal, and if there is at least one slot free, customers may return a bike by putting it into the free slot. To work well, a PBS has to have a reasonable density of stations in the covered region. Users can rent bikes at any station and return them at any other station.

PBS are mostly implemented in public-private partnership and are financed through advertisements on the bikes, subsidies from the municipalities, and subscription fees from the users. The costs for building and operating the system have to be covered. The problem of building or extending a PBS can in principle be seen as a facility location or hub location problem with network design aspects [33] and is not within the scope of this work.

For continuous operation of the system, besides maintaining the bikes and stations, providers in particular have to take care of *rebalancing* bikes among the stations such that users can rent and return bikes at any station with high probability. Stations should ideally neither run full nor empty, as these situations obviously significantly impact customer satisfaction.

Different approaches to achieve and maintain a reasonable balance exist. Most commonly, the PBS operator actively rebalances the stations by employing vehicles with trailers that pickup bikes at stations with excess of bikes and deliver them to stations with a lack of bikes. This is the scenario we will consider in the following, but there are also alternative approaches in which balance should be achieved by the users themselves [17, 37]. There, the operator provides incentives for their customers to rent bikes at stations with excess and to return them at stations with a lack of bikes. These incentives can be reduced subscription fees, prizes or discounts at special partners of the PBS. Both rebalancing strategies can also be used in conjunction.

The active rebalancing of a PBS by a vehicle fleet has in the literature been referred to as a *capacitated single commodity split pickup and delivery vehicle routing problem with multiple visits* [40]. Diverse variants of this problem, with different objectives and constraints, have already been considered, and different algorithmic approaches have been proposed, ranging from *mixed integer linear programming* (MIP) methods to metaheuristics and hybrids. To our knowledge, all these approaches allow for an arbitrary number of bikes to be picked up at some stations and delivered to other stations, just limited by the vehicles' and stations' capacities. Observations in practice, however, indicate that in a larger well-working bike sharing system it makes rarely sense to move only few bikes for rebalancing. Drivers actually almost always pickup a full vehicle load and deliver it completely to another station. Many stations even require several visits with full load pickups or deliveries. Due to budgetary reasons, typically only just enough drivers and vehicles are employed to achieve a reasonable balance most of the time, but basically never an ideal one where single bikes play a substantial role. Drivers should use their limited working time in a best way to optimize the PBS's overall balance as far as possible. The described scenario is particularly true in case of our collaboration partner Citybike Wien[1].

Following this observation, we investigate here a simplified problem definition in which *only full vehicle loads are considered* for movement among the rental stations. This restriction appears to have only a minor impact at the achieved quality of the rebalancing in practice but eases the modeling and algorithmic solving essentially.

For this new problem formulation, we then propose three exact solution approaches: a compact MIP model, a logic-based Benders decomposition (LBBD), and a variant thereof, namely Branch-and-Check (BAC). Moreover, we compare with previously proposed and leading metaheuristics allowing flexible numbers of picked up and delivered bikes, concluding that the restriction to only full vehicle loads affects the finally achieved balance in practical scenarios indeed in only minor ways.

This article is organized as follows: The next section presents the details of our new problem formulation and Section 3 summarizes related work. In Section 2.3 the compact MIP model is introduced, whereas Section 4 describes the LBBD and Section 4.3 its variant BAC. Computational results are shown in Section 6, and finally, we conclude in Section 7.

## 2. Problem Statement

We first summarize aspects of existing problem formulations for *Balancing Bike Sharing Systems* (BBSS) and then state our new approach, giving respective formal definitions.

Generally, previous works distinguish two types of problem variants for BBSS, namely the *static* and the *dynamic* case.

In the *static scenario* we are *given an initial state* of the system, i.e., initial fill levels for all stations, and a *desired target state* of the system, i.e., target fill levels or demand intervals for all stations.

[1]http://www.citybikewien.at

Technical Report AC-TR-15-004

Technical Report AC-TR-15-004

For the static case, a significant variety of different optimization goals has been considered in the literature, e.g., minimizing the traveling costs [6, 10] where balancing is modeled as a hard constraint, or minimizing the total number of expected shortages [41].

A quite challenging task is to determine best suited target fill levels for the optimization. This has to be done with caution because the final state at the end of rebalancing is the initial state for the next day(s) in the static model. The customer demand of renting and returning bikes is the crucial factor when target values for the rebalancing operations have to be determined. Thus, a sophisticated demand prognosis is necessary to estimate well-suited target values. Rudloff and Lackner [42] build such a prognosis model based on historical data of the system of Citybike Wien based on various impact factors like weather, day of the week, time of the day, temperature, etc. They also consider the influence of entirely full or empty neighboring stations. Han et al. [19] concentrate on the demand prediction for large-scale BSS. They describe the spatio-temporal correlation in BSS as an important factor for demand estimation. They verified their model on the record set they retrieved from the BSS *Vélib'* in Paris.

In general, the static problem variant neglects the dynamic interaction between the customers and the system as it does not consider the user demand during rebalancing and e.g., is appropriate for overnight rebalancing if the system is not in use during the night [41].

The *dynamic case* also considers user interactions during rebalancing. Only few works, however, exist in this direction. In [30] the user interactions and the demands are retrieved from historical data and implemented by a probabilistic model of Rudloff and Lackner [42], and the objective is to minimize *unsatisfied user demand* as well as to minimize deviation between initial and desired target fill levels. Contardo et al. [8] randomly generate demand values and try to minimize shortages and excesses of bikes over a prospective time horizon.

If the user demand is predicted reasonably well and the rebalancing takes place during the active times of the PBS, the dynamic case can thus in principle be more accurate than a static model but is also computationally much more demanding. Under the assumption that rebalancing should not primarily fulfill short-term needs and station capacities are reasonably large, static models are generally also accepted as a reasonably good approximation for systems where the rebalancing takes place during the operation hours. We therefore also concentrate on the static case here.

## 2.1. A BBSS Formulation Considering Full Vehicle Loads Only

As motivated already in the introduction, observations at *Citybike Wien* reveal that pickup and delivery of full vehicle loads clearly dominates practice. Due to economic reasons there is a financial limit on the labor costs, and rebalancing is done in such a way that a practically acceptable but usually not perfect balance of the stations' fill levels is achieved. Thus, the number of drivers respectively vehicles and their working times are a major limit, and the stations should be brought to specified target fill levels as far as possible, but reaching all of them exactly is (typically) out of question. The drivers are in principle daily faced with more work than can be feasibly done.

Technical Report AC-TR-15-004

Furthermore, many stations ideally require more than one, sometimes even several full vehicle loads to be delivered or picked up in order to achieve the desired target state. Most of the drivers' working time is consumed by traveling to the individual stations and parking somewhere nearby, however, required time for loading or unloading less or more bikes plays a comparably small role, and is frequently also neglected in existing models. In such a scenario, it becomes obvious that it is clearly most effective to move almost always approximately full vehicle loads from stations with a substantial excess of bikes to stations with a substantial demand.

Consequently, we assume in our new BBSS problem formulation that the vehicle is always either fully loaded with bikes or empty, dropping the consideration of moving only a certain number of bikes less than the vehicles full capacity. Concerning the objective function, our goal is to bring as many stations as far as possible to their specified target fill levels, respecting given working times, and the general constraints for feasible tours.

Considering only full vehicle loads simplifies existing models substantially. Typically, the consideration of the exact number of bikes to be moved requires an additionally embedded flow problem to be solved.

Of course, not dealing with partial vehicle loads comes along with a potential loss of accuracy, but the prediction of user demands which depends on, e.g., the weather, weekday, events in the stations' neighborhoods and the influence of neighboring stations involve in general uncertainties for the calculation of suitable target fill levels that can be safely assumed to dominate in practice.

## 2.2. Formal Problem Definition

We are given a set of stations $S$ and a set of homogeneous vehicles $L$. For the vehicles we are given a common capacity $Z$ and a common time budget $\hat{t}$ (drivers' shift times) within which the vehicles have to finish their routes. For each station $s \in S$ we are given the number of full vehicle loads $f_s$ to be delivered ($f_s \leq -1$) or picked up ($f_s \geq 1$) such that the station achieves its (approximately) ideal target fill level. Stations that are already at their desired target fill level (or require less than a full vehicle load) are ignored from any further consideration.

A station, to which bikes shall be delivered is called a *delivery station*, while a station from which bikes should be removed is called a *pickup station*. At pickup stations, only pickups may be performed, while at delivery stations, only deliveries, and we never allow more than $|f_s|$ visits at each station. Thus, a kind of buffering bikes at some station and moving them further later is explicitly excluded. Especially in our context with the consideration of full vehicle loads only, such solutions would not make sense anyway when the triangle inequality is fulfilled by the traveling times between stations, what can safely be assumed for practice.

For modeling tours with up to $|f_s|$ visits at each station $s \in S$, we define a directed bipartite graph $G = (V, A)$ as follows. Let $V_{\text{pic}} = \{(s,i) \mid s \in S \wedge f_s \geq 1,\ i = 1, \ldots, |f_s|\}$ be a set of nodes representing up to $|f_s|$ visits at each pickup station, and let $V_{\text{del}} = \{(s,i) \mid s \in S \wedge f_s \leq -1,\ i = 1, \ldots, f_s\}$ denote the respective potential visits at the delivery stations. $V = V_{\mathrm{pic}} \cup V_{\mathrm{del}}$ then refers to the joined set of all potential visits, and the arc set of graph $G$ is given by $A = \{(u, v), (v, u) \mid u \in V_{\mathrm{pic}}, v \in V_{\mathrm{del}}\}$.

We further extend the set of stations $V$ by two nodes 0 and $0'$ representing the depot at the beginning and the end of each tour, respectively, obtaining $V_0 = V \cup \{0, 0'\}$. Node 0 is connected to all pickup nodes, while $0'$ is connected to all delivery nodes, i.e., $A_0 = A \cup \{(0, v) \mid v \in V_{\mathrm{pic}}\} \cup \{(v, 0') \mid v \in V_{\mathrm{del}}\}$, yielding bipartite graph $G_0 = (V_0, A_0)$. We explicitly omit here an arc $(0, 0')$ which might be used for representing a vehicle that stays at the depot and does not do any station visits due to the fundamental assumption in our modeling that more than enough rebalancing work exists for keeping all vehicles busy.

Each arc $(u, v) \in A_0$ represents an actual trip from the location represented by visit $u$ to the location represented by visit $v$ and has a corresponding traveling time $t_{uv} > 0$ associated. This time also includes an estimated time for parking at the destination and in case of $v \neq 0'$ for handling the station's electronic system and for loading or unloading the bikes.

A solution to our problem is a set of $|L|$ simple paths in $G_0$ from node 0 to node $0'$ visiting all vertices in their vehicle's $l \in L$ corresponding subgraph. Let $r_l = (r_l^1, r_l^2, \ldots, r_l^{\rho_l})$ be the successive station visits in the route of vehicle $l \in L$, with $\rho_l$ being the number of visits and $V(r_l)$ corresponding to the set of station visits contained in route $r_l$. Due to the bipartite structure of $G_0$, as long as the path is not empty ($\rho_l > 0$) each odd stop must be performed at a pickup station, i.e., $r_l^1, r_l^3, \ldots, r_l^{\rho_l - 1} \in V_{\mathrm{pic}}$, while each even stop takes place at a delivery station, i.e., $r_l^2, r_l^4, \ldots, r_l^{\rho_l} \in V_{\mathrm{del}}$, and $\rho_l$ always is even.

A non-empty route $r_l$ is feasible with respect to the time budget $\hat{t}$ iff

$$t_{0 r_l^1} + \sum_{i=1}^{\rho_l - 1} t_{r_l^i r_l^{i+1}} + t_{r_l^{\rho_l} 0'} \;\leq\; \hat{t}. \tag{1}$$

By assumption all vehicles start empty at the beginning and have to return empty, which is implicitly guaranteed again by the bipartite graph.

By above definitions, we reduce the BBSS problem as introduced in [40] to a *selective unit-capacity one-commodity pickup and delivery problem with time budgets on a bipartite graph.*

As optimization goal, we consider in this work the maximization of the total number of station visits

$$\max \sum_{l \in L} \rho_l, \tag{2}$$

which corresponds to twice the number of moved full vehicle loads. By this objective function, we also minimize the sum of the deviations from the stations' target fill levels after the rebalancing, which is

$$\min \sum_{s \in S} |f_s| - \sum_{l \in L} \rho_l \tag{3}$$

Technical Report AC-TR-15-004

and is the primary objective of previous work such as [11, 12, 25, 40, 41]. In [40] the objective is particularly given as follows:

$$\min\ \omega^{\text{bal}} \sum_{s\in S} \delta_s + \omega^{\text{load}} \sum_{l\in L} \sum_{i=1}^{\rho_l} |y^i_{l,r^i_l}| + \omega^{\text{work}} \sum_{l\in L} t_l, \tag{4}$$

where $\omega^{\text{bal}}, \omega^{\text{load}}, \omega^{\text{work}}$ are weighting factors, $\delta_s = |a_s - q_s|$ denotes the deviation of the final fill level $a_s$ from the target fill level $q_s$ at station $s \in S$, $y^i_{l,r^i_l}$ denotes the number of bikes loaded ($> 0$) or unloaded ($< 0$) by vehicle $l \in L$ at station $r^i_l$, and $t_l$ is the total working time of vehicle $l$.

**Proposition 1.** *When considering only balance optimization, the objective functions shown in equation (3) and (4) correspond to each other (except for rounding errors resulting from the fact that we now only consider full vehicle loads).*

*Proof.* As we only focus on the balance aspect here, i.e., minimizing the deviation between final and target fill levels, we set the weighting factors in equation (4) to $\omega^{\text{bal}} = 1$, $\omega^{\text{load}} = 0$, $\omega^{\text{work}} = 0$, effectively ignoring the second and third term. Equation (4) can then be rewritten as

$$\min \sum_{s\in S} |a_s - q_s| = \min \sum_{s\in S_{\text{pic}}} a_s - q_s + \sum_{s\in S_{\text{del}}} q_s - a_s. \tag{5}$$

Let $p_s$ be the initial fill level for station $s \in S$, then in a static context, the final fill level can also be expressed as $a_s = p_s - \sum_{l\in L} \sum_{i=1}^{\rho_l} y^i_{l,r^i_l}$ which results in

$$\min \sum_{s\in S_{\text{pic}}} \left( p_s - q_s - \sum_{l\in L} \sum_{i=1}^{\rho_l} y^i_{l,s} \right) + \sum_{s\in S_{\text{del}}} \left( q_s - p_s + \sum_{l\in L} \sum_{i=1}^{\rho_l} y^i_{l,s} \right). \tag{6}$$

To show the correspondence of equation (3) to equation (6), equation (3) is multiplied by the vehicle capacity $Z$ such that the deviation in full vehicle loads is transformed to the actual deviation in the number of bikes

$$\min\ Z \cdot \left( \sum_{s\in S} |f_s| - \sum_{l\in L} \rho_l \right) \tag{7}$$

Required full loads to balance station $s \in S$ can be calculated as follows

$$f_s = \left\lceil \frac{p_s - q_s}{Z} \right\rceil \forall s \in S_{\text{pic}}, \quad \text{and} \quad f_s = \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor \forall s \in S_{\text{del}}, \tag{8}$$

which can be used in equation (7) to get

$$\min\ Z \cdot \left( \sum_{s\in S_{\text{pic}}} \left\lceil \frac{p_s - q_s}{Z} \right\rceil - \sum_{s\in S_{\text{del}}} \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor - \sum_{l\in L} \rho_l \right). \tag{9}$$

Technical Report AC-TR-15-004

Let $b_s = \sum_{l \in L} |\{r_l^i \mid r_l^i \in r_l, r_l^i = s\}| \; \forall s \in S$ the number of full vehicle loads delivered to or picked up at station $s \in S$, then we can rewrite equation (9) as

$$\begin{aligned}
&\min \; Z \cdot \left( \sum_{s \in S_{\text{pic}}} \left\lceil \frac{p_s - q_s}{Z} \right\rceil - \sum_{s \in S_{\text{del}}} \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor - \sum_{s \in S} b_s \right) = \\
&\min \; Z \cdot \left( \sum_{s \in S_{\text{pic}}} \left\lceil \frac{p_s - q_s}{Z} \right\rceil - \sum_{s \in S_{\text{del}}} \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor - \sum_{s \in S_{\text{del}}} b_s - \sum_{s \in S_{\text{pic}}} b_s \right) = \\
&\min \; Z \cdot \left( \sum_{s \in S_{\text{pic}}} \left( \left\lceil \frac{p_s - q_s}{Z} \right\rceil - b_s \right) - \sum_{s \in S_{\text{del}}} \left( \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor - b_s \right) \right) = \\
&\min \sum_{s \in S_{\text{pic}}} \left( Z \cdot \left\lceil \frac{p_s - q_s}{Z} \right\rceil - Z \cdot b_s \right) + \sum_{s \in S_{\text{del}}} \left( Z \cdot \left\lceil \frac{q_s - p_s}{Z} \right\rceil + Z \cdot b_s \right)
\end{aligned} \tag{10}$$

Comparing equation (6) with equation (10) shows that the terms $Z \cdot b_s$ and $\sum_{l \in L} \sum_{i=1}^{\rho_l} y_{l,s}^i$ correspond to each other as both represent the number of moved bikes by the vehicles in the system. Moreover, the terms $Z \cdot \left\lceil \frac{p_s - q_s}{Z} \right\rceil, Z \cdot \left\lceil \frac{q_s - p_s}{Z} \right\rceil$ and $p_s - q_s, q_s - p_s$ correspond to each other except for rounding errors due to the consideration of only full vehicle loads. □

## 2.3. Compact Mixed Integer Linear Programming Model

We now formulate the problem as a compact MIP model using *assignment variables* $x_{vl} \in \{0, 1\}$ to state the assignment of station visits $v \in V$ to vehicles $l \in L$ and *arc selection variables* $y_{uv}^l \in \{0, 1\}$ to describe the tour for each vehicle. Subtours are eliminated via Miller-Tucker-Zemlin inequalities [34] utilizing further continuous variables $a_v$ for the nodes $v \in V$.

$$\max \quad \sum_{l \in L} \sum_{v \in V} x_{vl} \tag{11}$$

$$\text{s.t.} \quad \sum_{l \in L} x_{vl} \leq 1 \qquad \forall v \in V \tag{12}$$

$$\sum_{v \in V_{\text{pic}}} x_{vl} = \sum_{v \in V_{\text{del}}} x_{vl} \qquad \forall l \in L \tag{13}$$

$$\sum_{l' \in L} x_{(s,i)l'} \geq x_{(s,i+1)l} \qquad \forall s \in S, \; l \in L, \; i = 1, \ldots, f_s - 1 \tag{14}$$

$$\sum_{v \in V_{\text{pic}}} y_{0v}^l = 1 \qquad \forall l \in L \tag{15}$$

$$\sum_{v \in V_{\text{del}}} y_{v0'}^l = 1 \qquad \forall l \in L \tag{16}$$

$$\sum_{(u,v) \in A_0} y_{uv}^l = x_{ul} \qquad \forall l \in L, \; u \in V \tag{17}$$

Technical Report AC-TR-15-004

$$\sum_{(u,v)\in A_0} y^l_{uv} = x_{vl} \qquad \forall l \in L,\ v \in V \tag{18}$$

$$\sum_{(u,v)\in A_0} y^l_{uv} = \sum_{(v,u)\in A_0} y^l_{vu} \qquad \forall l \in L,\ v \in V \tag{19}$$

$$a_u - a_v + |V| \cdot y^l_{uv} \leq |V| - 1 \qquad \forall l \in L,\ (u,v) \in A \tag{20}$$

$$\sum_{(u,v)\in A_0} t_{uv} \cdot y^l_{uv} \leq \hat{t} \qquad \forall l \in L \tag{21}$$

$$x_{vl} \in \{0,1\} \qquad \forall l \in L,\ v \in V \tag{22}$$

$$y^l_{uv} \in \{0,1\} \qquad \forall l \in L,\ (u,v) \in A_0 \tag{23}$$

$$1 \leq a_v \leq |V|, a_v \in \mathbb{R} \qquad \forall v \in V \tag{24}$$

The objective function (11) maximizes the number of full vehicle loads picked up and delivered to the stations and thus, the total balance increase in the PBS, compare equation (3). Inequalities (12) state that every station visit is performed by at most one vehicle. By equalities (13) we explicitly define that every tour contains the same amount of pickup visits as delivery visits. Note that these equations are in principle redundant but we include them nevertheless as they might be helpful from a computational point of view. Inequalities (14) are used for symmetry breaking among the visits of the same station: The $i+1$-th visit can only be performed when the $i$-th visit is performed, for $i = 1, \ldots, f_s - 1$ and each station $s \in S$. For each vehicle the depot's starting node 0 has to have one outgoing arc (15), and similarly, the depot's target node $0'$ has to have one incoming arc (16). The arc selection variables are linked with the assignment variables as follows: Equalities (17) ensure that every node $u \in V$ has one outgoing arc iff it is assigned to vehicle $l$, i.e., $x_{ul} = 1$, while equalities (18) guarantee that each node $v \in V$ which is assigned to vehicle $l \in L$ has to have one corresponding ingoing arc. Equalities (19) express that the number of ingoing arcs has to be equal to the number of outgoing arcs for each node $v \in V, l \in L$. We eliminate subtours by inequalities (20) by computing an ordering of the nodes in variables $a_v$. Inequalities (21) guarantee that the routes for each vehicle lie within the allowed time budget $\hat{t}$. Finally, (22) to (24) define the domains of the decision variables.

For small instances, a state-of-the-art MIP solver such as CPLEX is able to directly yield proven optimal solutions by this model in reasonable time, see the experimental results in Section 6. The approach, however, does not scale well to larger instances.

## 3. Related Work

In this section we give an overview on existing algorithmic approaches for finding reasonable routes for balancing PBSs and other problems related to our simplified problem formulation considering full vehicle loads only.

As already pointed out in the above section, essentially all existing models for rebalancing PBSs consider flexible numbers of bikes to be loaded or unloaded at each visit, and most work addresses the static case only. Several different problem variants with

Technical Report AC-TR-15-004

Technical Report AC-TR-15-004

different objectives and side constraints exist, and different solution approaches have been proposed for them. Direct comparisons are therefore quite hard. Many of the described approaches rely on MIP techniques, but there also exist (meta-)heuristics and hybrid metaheuristics, which appear to be particularly well suited for larger scenarios.

Before starting with the literature review it should be pointed out that an overview paper about *shared mobility systems* has been published by Laporte et al. [32] containing chapters about rebalancing incentives and vehicle repositioning approaches.

## 3.1. MIP Approaches

Chemla et al. [6] proposed an exact branch-and-cut approach for the single-vehicle case considering it a hard constraint to exactly reach all given target fill levels. The approach is based on a relaxed MIP model yielding a lower bound and a tabu search for obtaining heuristic solutions and thus upper bounds.

Raviv et al. [41] proposed several MIP models minimizing user dissatisfaction and operational costs. These include a time-indexed as well as an arc-indexed formulation which is restricted in the sense that a station may only be visited once by the same vehicle. They also incorporate loading and unloading times proportional to the number of bikes moved. By additionally applying algorithmic enhancements to their MIP models they are able to solve instances up to 60 stations with reasonable optimality gaps.

Schuijbroek et al. [44] describe approaches for determining service level requirements at the stations and vehicle routes for the rebalancing at the same time. An initial MIP model turns out to be intractable for instances of practical size. Consequently, the authors derive a cluster-first route-second heuristic where they first assign stations to clusters by a MIP model and then they solve an independent vehicle routing problem (VRP) for each cluster. In our approach, we will follow a similar basic idea for decomposition, but extend it to an exact LBBD.

Similarly to Schuijbroek et al., Erdoğan et al. [13] define demand intervals for each station. They consider only the single-vehicle case and aim at minimizing traveling costs for the vehicle and handling costs for the rebalanced bikes. Erdoğan et al. present a branch-and-cut formulation, apply valid inequalities from the VRP literature and also present a Benders decomposition scheme. Their approaches solve instances up to 50 stations to optimality.

## 3.2. (Meta-)Heuristics and Hybrid Approaches

Due to the practical complexity of BBSS, (meta-)heuristics appear also particularly meaningful especially for larger systems. Diverse metaheuristic approaches are described in the literature. Rainer-Harbach et al. [39] introduced a greedy construction heuristic (GCH) and a variable neighborhood search (VNS) with an embedded variable neighborhood descent. These methods have been tested for instances with up to 700 stations, for which they provided very reasonable results. Papazek et al. [36] have developed a pilot heuristic [50] which improved the GCH from [39] significantly, a greedy randomized adaptive search procedure (GRASP) upon both construction

Technical Report AC-TR-15-004

heuristics, performing very well on instances with a high number of rental stations. Raidl et al. [38] examined different strategies for determining optimal loading and unloading decisions for given routes within a metaheuristic by specialized maximum-flow and linear programming approaches. Rainer-Harbach et al. [40] refined their work on metaheuristics for the static case by providing comprehensive computational tests and have also introduced their time-indexed and hop-indexed MIP models. Papazek et al. [35] investigated diverse path relinking extensions for GRASP.

The dynamic case was considered by Kloimüllner et al. [30], who proposed a problem model in which flexible demand functions in dependence of time can be considered for all the stations. By separating the demand functions into continuous monotonic pieces and dealing with them appropriately, a complete discretization of time could be avoided. As solution approaches, the authors extended the GRASP and VNS metaheuristics from [40]. The VNS was able to solve instances with up to 90 stations reasonably well.

Di Gaspero et al. further describe a constraint programming approach [11] and a hybridization of it with ant colony optimization [12]. They tested on the same benchmark set as Rainer-Harbach et al. [40]. Although the hybrid ant colony optimization performed better than the pure constraint programming, these methods were not able to yield competitive results.

Vogel et al. [49] propose a MIP model for the resource allocation problem arising in PBSs. They aim at minimizing the traveling costs as well as the handling costs for the relocated bikes. Furthermore, they add a penalty to the objective function for missing bikes and missing free slots at the stations. As for real-world instances the size of the MIP model is too large to be solved directly, the authors suggest a MIP-based large neighborhood search following a fix-and-optimize strategy.

Forma et al. [15] propose the following 3-step hybrid metaheuristic. First, stations are clustered according to geographical data and initial inventory by using a savings heuristic. In a second step, it is decided which vehicle visits which clusters of stations by using a revised MIP model originally stated in [41]. Vehicles are allowed to visit multiple clusters but one cluster is assigned to exactly one vehicle. In a third step, routing problems are solved for each cluster independently. The authors report results for instances with up to 200 stations and three vehicles.

## 3.3. Other Related Problems and Approaches

Obviously, our simplified BBSS model in which only full vehicle loads are considered is related to diverse other vehicle routing and in particular pickup and delivery problems. There are, however, several special aspects that need to be considered by a meaningful solution approach, in particular that not all stations need to be visited, that a time budget is given, and that tours are sought on a bipartite graph.

A similar problem occurs in the domain of waste collection, for which Aringhieri et al. [2] describe a GRASP and a tabu search. In this problem there is also given a bipartite graph resulting in alternating tours between pickup and delivery places. However, multiple commodities representing different types of waste are considered there. The objective is to reduce the number of tours needed to dispose all the

Technical Report AC-TR-15-004

waste and thus, collecting all the waste is considered here as hard constraint, whereas we aim to optimize the quantity of moved commodity within the given time budget.

Another problem related to the one introduced here is the one-commodity full-truckload pickup and delivery problem (1-FTPDP) proposed by Gendreau et al. [18]. This is a variant of the well-known pickup and delivery problem where a truck has to alternatively visit pickup as well as delivery customers all demanding a unit-capacity pickup respectively delivery. In contrast to our problem the supply and demand of each customer has to be satisfied. Thus, the authors add copies of the depot either to the set of pickup customers or to the set of delivery customers to ensure enough supply respectively demand of the customers. There are no time-budget constraints and all customers have to be visited exactly once. The authors model the problem by solving a routing problem through the set of pickup customers and then, assign the delivery customers to the pickup customers. The problem is solved to optimality by relying on classical and generalized Benders decomposition. They also present a traveling salesman problem (TSP) formulation of the problem based on classical subtour elimination constraints. Starting with an initial empty set of subtour elimination constraints they separate them by detecting all connected components and adding subtour elimination constraints for them accordingly. They compare their two approaches based on classical and generalized Benders decomposition with their TSP formulation and conclude that the TSP formulation outperforms the classical as well as the generalized Benders decomposition, although the authors note that there is room for improvement of the approaches based on Benders decomposition.

Related to our problem formulation also is the one-commodity pickup and delivery traveling salesman problem (1-PDTSP) described by Hernández-Pérez et al. [21, 22, 23, 24], and the selective pickup and delivery problem (SPDP) studied by Ting and Liao [48]. In the 1-PDTSP a depot and several customers are given which are either pickup or delivery customers and the aim is to find a minimum distance route visiting all customers starting and ending at the depot and satisfying all the supplies and demands. In addition, Salazar-González and Santos-Hernández [43] introduce the split-demand one-commodity pickup and delivery traveling salesman problem where a truck has to visit a number of delivery and pickup customers multiple times respecting a maximum number of visits per customer. Also the depot may be visited multiple times. However, they do not consider time-budget constraints and all demands have to be fulfilled. They propose an exact model which is solved by Benders decomposition where the separation problem is modeled as a maximum-flow problem. They report interesting and excellent results on an extensive set of benchmark instances. In the SPDP not all pickup nodes have to be visited, but all delivery demands need to be fulfilled. Moreover, somewhat related also is the prize collecting traveling salesman problem introduced by Balas [3], in which a prize is paid for every visited city and/or a penalty has to be paid for each city which is not visited. A minimum prize money has to be earned, and the objective is to minimize the routing costs as well as the penalty incurred by cities which have not been visited.

Especially when considering our decomposition approach which will be described in Section 4, we obtain as subproblems independent Hamiltonian path problems for the individual vehicles. These problems can be modeled as classical asymmetric TSPs

(ATSP) on bipartite graphs. Concerning this special TSP variant, not much specific work exists. To the best of our knowledge, Frank et al. [16] have been the first researchers considering bipartite, symmetric TSPs for which they proposed a 2-factor approximation algorithm. Srivastav et al. [46] analyzed the problem of finding tours for pick-and-place robots which showed up of consisting of a an assignment problem and a bipartite TSP. Given an initial bin assignment the authors proposed several approximation algorithms for the bipartite TSP. Further work on approximation algorithms for the bipartite TSP was done by Baltz and Srivastav [4] as well as Shurbevski et al. [45]. However, these algorithms are more of theoretical interest. We will apply the well-known *Concorde* TSP solver [1, 9] to tackle these subproblems, not further exploiting the underlying bipartite graph structure.

# 4. Logic-based Benders decomposition

We first introduce the term *LBBD* and then describe the application of an LBBD scheme to BBSS.

## 4.1. Introduction

In 1962 Benders came up with his classical decomposition technique to solve large MIP problems [5]. This approach is in principle applicable if the problem can be split into a master problem making use of only a subset of the variables including the complicating integer variables, and an easier subproblem on the remaining continuous variables when the master problem variables are assumed to be fixed to certain values. The solution approach iterates by solving master problem instances and subproblems. After the master problem is solved, a corresponding subproblem is obtained by fixing the master problem's variables in the original formulation to the obtained values. From the solution of the subproblem's *linear programming (LP) dual* one derives feasibility and/or optimality cuts which are added to the master problem in each iteration. The whole process is repeated until no further Benders cuts can be derived and an optimal solution has been obtained.

Erdoğan et al. [13] propose a Benders decomposition scheme for solving the static rebalancing problem arising in BSS. When applying Benders decomposition to VRPs often the master problem, containing the complicating variables, is hard to solve. Thus, Lai et al. [31] came up with a hybrid of Benders decomposition and a genetic algorithm (GA). They solve the master problem by the GA and the subproblems via a MIP model by a commercial solver.

LBBD generalizes classical Benders decomposition by also allowing integer variables or even nonlinearities in the subproblem. This is achieved by replacing the LP dual by a more general concept called *inference dual* [27]. Typically, Benders cuts are here obtained via logical deduction. In several applications, in particular in the domain of scheduling, LBBD achieved remarkable results.

Hooker [26] presents a solution method applicable to generic scheduling problems where he models the master problem as a MIP and solves the subproblems by con-

Technical Report AC-TR-15-004

straint programming (CP). Reported results on the LBBD outperform a pure MIP and and a pure CP approach. Harjunkoski and Grossmann [20] propose a decomposition approach for multistage scheduling problems. The master problem, an assignment problem, is modeled as a MIP whereas for the subproblems they employ two strategies for feasibility checking: One which utilizes a CP approach and another one where a MIP model is used for the feasibility check. They have shown that the hybrid decomposition approach by solving the master problem as a MIP and the subproblems with their CP approach has been superior to a pure MIP or pure CP approach. Furthermore, solving the subproblem, the sequencing of jobs, with the CP approach has been superior to the feasibility check by the MIP.

There are two types of Benders cuts, namely, infeasibility cuts and optimality cuts. Infeasibility cuts state that the current master solution is not feasible and avoid its generation in future iterations, whereas optimality cuts provide new bounds on the objective value for the current master problem solution. In every iteration except the last, one or more cuts are generated where every single cut reduces the master problem's search space, or more precisely its underlying LP polytope – the more the better in general. Thus, it should also be considered to strengthen obtained Benders cuts as far as possible, which is especially in case of the LBBD frequently done by heuristics or by constraint programming techniques, cf. the greedy algorithms proposed by Hooker [26].

We note that a technique which is similar to the principles of LBBD is called *combinatorial Benders cuts*, cf. Codato and Fischetti [7].

Technical Report AC-TR-15-004

## 4.2. Application to BBSS

The problem consists of an assignment problem (AP) and multiple Hamiltonian path problems with time budgets that are interconnected. The AP is given in the proposed model by equations (12)–(14), the Hamiltonian path problems are represented by equations (19)–(21), and the connections between the AP and Hamiltonian path problems are given by equation (17) and (18). In the following we decompose the problem correspondingly by applying *LBBD*. In this approach, we iteratively solve a master problem, corresponding to the AP, and subproblems corresponding to the Hamiltonian path problems but are modeled as ATSPs. The solutions of the subproblems will yield Benders infeasibility cuts for restricting the master problem in the further iterations. The following section discusses this decomposition approach in detail.

In the following we show how LBBD is applied to our MIP for BBSS. Section 4.2.1 describes the master problem and states its MIP formulation, while Section 4.2.2 discusses the subproblem and proposes a corresponding solution approach. Section 4.2.3 shows how the master problem and subproblem interact and how the algorithm finally yields an optimal solution. Section 4.3 introduces the alternative to LBBD, namely BAC.

### 4.2.1. Master problem

We decompose the model (11)–(24) from Section 2.3 by focusing in the master problem on the clustering aspect, i.e., the AP, yielding multiple Hamiltonian path problems as subproblems. Our method was inspired by the cluster-first route-second method introduced by Fisher and Jaikumar [14] and also applied for BBSS by Schuijbroek et al. [44].

In order to strengthen the master problem such that a relatively meaningful clustering is determined from the beginning on, it is crucial to estimate the route durations for the cluster and exclude clusters that obviously cannot be handled by a single vehicle. Hooker [26] also reveals that it is important, for the success of the approach, to include a *relaxation of the subproblem within the master problem.* Ideally, this route duration estimation should come close to the real minimal Hamiltonian path durations and introduce only a reasonable overhead in the master problem's model. However, it is important that the determined approximate trip durations are guaranteed lower bounds for the real durations, as otherwise sets of station visits might be excluded from becoming clusters, despite feasible routes would actually exist for them.

A lower bound for a TSP that can relatively easily be expressed by a linear program is obtained from the minimum spanning tree relaxation of the TSP. As we can model the Hamiltonian path problem as an ATSP, we relax the problem of finding an optimal ATSP tour to the *minimum* 0*-arborescence problem*, i.e., a minimum, from the depot outgoing, arborescence.

The MIP formulation of our master problem primarily uses the assignment variables $x_{vl},\ v \in V,\ l \in L$ from the original problem. For determining the lower bounds for the vehicles' tour durations via the arborescence polytope, flow variables $f^l_{uv}$ and arc selection variables $y^l_{uv} \in \{0,1\}$ for all vehicles $l \in L$ and arcs $(u,v) \in A_0$ are used.

Furthermore, we define $\beta$ to be an upper bound on the maximal number of station visits per vehicle. This upper bound is derived by solving the single-vehicle case of the problem, for which the MIP model is given in Appendix A. This single vehicle case is in practice much easier to solve than our complete problem. In our test discussed in Section 6, we typically obtained optimal solutions within seconds, and stopped the solving after a CPU-time limit of 5min and then took the obtained rounded down upper bound to the optimal solution value as $\beta$.

Given these decision variables, preprocessing values and parameters, the *master problem* (MP) is stated as follows:

$$\max \quad \sum_{l \in L} \sum_{v \in V} x_{vl} \tag{25}$$

$$\text{s.t.} \quad \sum_{v \in V} x_{vl} \le \beta \qquad \forall l \in L \tag{26}$$

$$\sum_{l \in L} x_{vl} \le 1 \qquad \forall v \in V \tag{27}$$

$$\sum_{v \in V_{\text{pic}}} x_{vl} = \sum_{v \in V_{\text{del}}} x_{vl} \qquad \forall l \in L \tag{28}$$

Technical Report AC-TR-15-004

$$\sum_{l' \in L} x_{(s,i)l'} \geq x_{(s,i+1)l} \qquad \forall s \in S,\ l \in L,\ i = 1, \dots, f_s - 1 \tag{29}$$

$$\sum_{(0,v) \in A_0} y^l_{0v} = 1 \qquad \forall l \in L \tag{30}$$

$$\sum_{(u,0') \in A_0} y^l_{u0'} = 1 \qquad \forall l \in L \tag{31}$$

$$y^l_{uv} \leq x_{ul} \qquad \forall l \in L,\ u \in V,\ (u,v) \in A_0 \tag{32}$$

$$y^l_{uv} \leq x_{vl} \qquad \forall l \in L,\ v \in V,\ (u,v) \in A_0 \tag{33}$$

$$\sum_{(0,v) \in A_0} f^l_{0v} = \sum_{v \in V} x_{vl} + 1 \qquad \forall l \in L \tag{34}$$

$$\sum_{(v,0') \in A_0} f^l_{v0'} = 1 \qquad \forall l \in L \tag{35}$$

$$\sum_{(u,v) \in A_0} f^l_{uv} - \sum_{(v,w) \in A_0} f^l_{vw} = x_{vl} \qquad \forall l \in L,\ v \in V \tag{36}$$

$$f^l_{uv} \leq \begin{cases} (\beta + 1) \cdot y^l_{0v} & \text{if } u = 0 \\ \beta \cdot y^l_{uv} & \text{if } v \in V_{\text{pic}} \\ (\beta - 1) \cdot y^l_{uv} & \text{else} \end{cases} \qquad \forall l \in L,\ (u,v) \in A_0 \tag{37}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = \sum_{v \in V} x_{vl} + 1 \qquad \forall l \in L \tag{38}$$

$$\sum_{(u,v) \in A_0} t_{uv} \cdot y^l_{uv} \leq \hat{t} \qquad \forall l \in L \tag{39}$$

$$x_{vl} \in \{0,1\} \qquad \forall l \in L,\ v \in V \tag{40}$$

$$y^l_{uv} \in \{0,1\} \qquad \forall l \in L,\ (u,v) \in A_0 \tag{41}$$

$$f^l_{uv} \in \mathbb{R}^+ \qquad \forall l \in L,\ (u,v) \in A_0 \tag{42}$$

As in the compact model, the objective function (25) to be maximized is the total number of performed station visits. The maximum number of station visits per vehicle are bounded upwards by $\beta$ (26), the optimal solution or rounded down upper bound of the single-vehicle case, cf. Appendix A. Inequalities (27) state that any station visit can only be performed by at most one vehicle. Equations (28) ensure that for every vehicle the number of assigned delivery station visits corresponds to the number of assigned pickup station visits. Inequalities (29) ensure that the $i+1$-th visit of a station can only be performed when an $i$-th visit takes place. Equalities (30) and (31) state that each vehicle leaves node 0 once and arrives at $0'$ once, respectively. Assignment variables $x_{vl}$ are linked with the arc selection variables $y^l_{uv}$ by inequalities (32) and (33). It is ensured that an arc $(u,v)$ can only be used in the arborescence if both $u \in V$ and $v \in V$ are assigned to vehicle $l$. Note that these inequalities are in principle redundant because it is also implicated by the constraints for the flow conservation but we include them nevertheless as they might be helpful from a computational point of view.

Technical Report AC-TR-15-004

Technical Report AC-TR-15-004

The arborescence is realized by the single commodity flow conservation equations (34)–(38). According to (34) the amount of flow sent out from the depot at node 0 corresponds to the number of nodes assigned to vehicle $l$ plus one to also reach $0'$, i.e., to get back to the depot. The consumption of this last unit of flow at $0'$ is ensured by (35). Equalities (36) provide the flow conservation for all station visits $v \in V$, where one unit of flow is consumed by each station visit assigned to vehicle $l \in L$. Inequalities (37) link the flow variables with the arc selection variables $y^l_{uv}$, i.e., a positive flow may only occur on a selected arc. Equations (38) state for each arborescence that the total number of arcs must be one more than the total number of nodes, i.e., station visits assigned to vehicle $l \in L$. Inequalities (39) ensure that for each vehicle the approximated routing durations, i.e., the total times of the arborescence, lie within the allowed time budget $\hat{t}$. Finally (40)–(42) are the domain definitions of the decision variables. Variables $x_{vl}$ and $y^l_{uv}$ are binary whereas the flow variables $f^l_{uv}$ are continuous.

### 4.2.2. Subproblems

A solution to the master problem yields an assignment of stations to vehicles in variables $x_{vl}$. Let $G_l = (V_l, A_l)$ with node set $V_l = \{v \mid v \in V, x_{vl} = 1\}$ and arc set $A_l = \{(u,v) \mid (u,v) \in A, x_{ul} = 1, x_{vl} = 1\} \cup \{(0,v) \mid v \in V^{\text{pic}}, x_{vl} = 1\} \cup \{(v,0) \mid v \in V^{\text{del}}, x_{vl} = 1\}$ be the corresponding subgraph for vehicle $l \in L$. The *subproblem* (SP) in our LBBD corresponds then to the task of finding for each vehicle $l \in L$ in its corresponding subgraph $G_l$ a Hamiltonian path from 0 to $0'$ visiting each node $v \in V_l \cup \{0, 0'\}$ exactly once and having a total duration that does not exceed $\hat{t}$. Thus, our Benders subproblem decomposes into $|L|$ independent Hamiltonian path problems that are essentially decision variants of the ATSP, when considering that nodes 0 and $0'$ actually represent the same depot and might be further connected with an arc $(0', 0)$.

As sophisticated solvers for the TSP exist, we utilize one of them in our solution approach instead of implementing one on our own: *Concorde* [1, 9] is a state-of-the-art TSP solver for the symmetric traveling salesman problem (STSP) on complete graphs. We convert each of our directed ATSP instances into an STSP instance by employing the 2-node transformation described by Jonker and Volgenant [28, 29]. A symmetric auxiliary graph $G^{\text{aux}} = (V^{\text{aux}}, E^{\text{aux}})$ with associated costs $t^{\text{aux}} : E^{\text{aux}} \to \mathcal{R}^+$ is derived. Its set of vertices consists of two nodes for each one in $V_l$ and two nodes 0 and $0'$ representing the depot: $V^{\text{aux}} = \{v \mid v \in V_l\} \cup \{v' \mid v \in V_l\} \cup \{0, 0'\}$. As Concorde works on a complete graph, we set $E^{\text{aux}} = V^{\text{aux}} \times V^{\text{aux}}$ and define the edge costs as follows:

$$t^{\text{aux}}_{vv'} = 0 \qquad \forall v \in V_l \tag{43}$$

$$t^{\text{aux}}_{uv} = t^{\text{aux}}_{u'v'} = \infty \qquad \forall u, v \in V_l,\ u \neq v \tag{44}$$

$$t^{\text{aux}}_{uv'} = t_{uv} + M \qquad \forall (u,v) \in A_l \tag{45}$$

$$t^{\text{aux}}_{uv'} = \infty \qquad \forall u, v \in V_l,\ u \neq v,\ (u,v) \notin A_l \tag{46}$$

Figure 1 shows the derivation of the auxiliary graph on an example. Note that the big-M is needed to ensure that the zero-cost edges between all nodes and their duplicates

Technical Report AC-TR-15-004

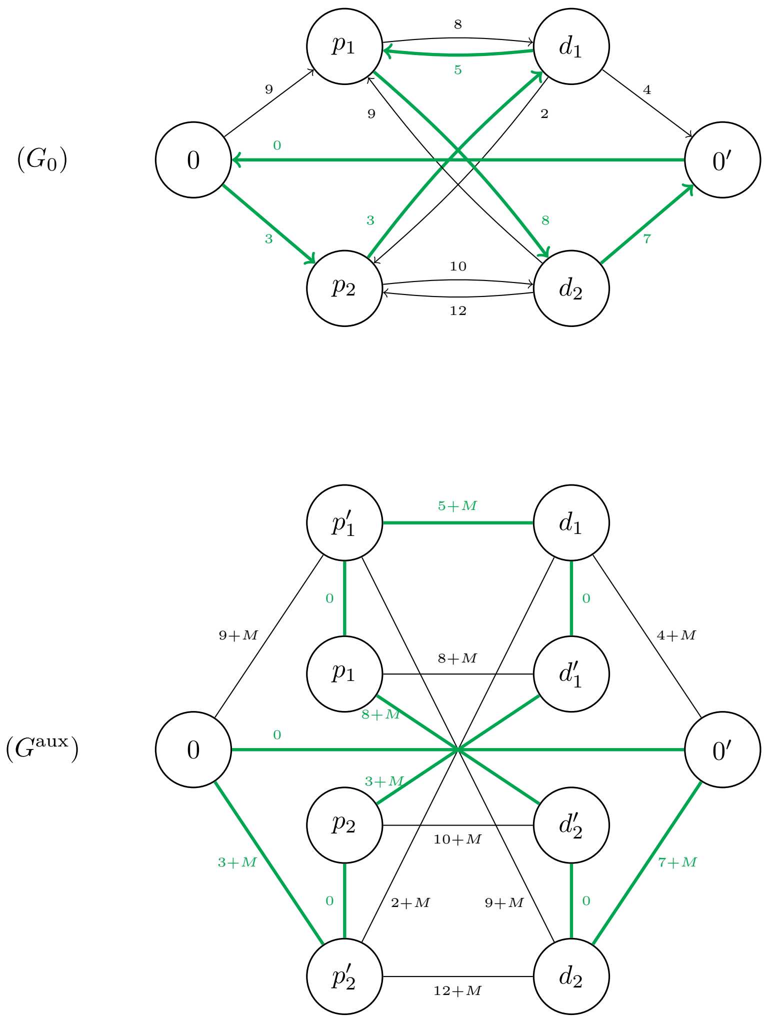

Figure 1: An example for the conversion of our subproblem on subgraph $G_l$ into a symmetric traveling salesman problem instance on an auxiliary graph $G^{\text{aux}}$. Pickup stations are referred by $p_1$ and $p_2$, $d_1$ and $d_2$ denote delivery stations, 0 is the depot and $0'$ is the copy of the depot. Note, that $G^{\text{aux}}$ actually is a complete graph. However, infeasible edges with $t_{uv} = \infty$, $\forall (u, v) \in G^{\text{aux}}$ are omitted for the sake of readability. The optimal solution in $G^{\text{aux}}$ and the corresponding optimal solution in $G_l$ are drawn as bold, green edges respectively arcs.

$(v, v')$ are always used in an optimal solution for the converted STSP. Thus, it has to be ensured that the big-M constant is large enough such that this property is guaranteed.

**Proposition 2.** *There is a one-to-one correspondence between optimal solutions to the converted STSP with finite objective and optimal solutions for the ATSP.*

*Proof.* Let $\mathcal{C}$ be the set of all Hamiltonian cycles in $G^{\text{aux}}$ which contain $(v, v')$ for all $v \in V_l$ and $C^{\text{aux}} \in \mathcal{C}$. We define the following corresponding subgraph of $G_l$, for which we will prove that it is a Hamiltonian cycle:

$$C = \{(u, v) \mid u, v' \in C^{\text{aux}}, u \neq v\} \cup \{(v, u) \mid u', v \in C^{\text{aux}}, u \neq v\}.$$

Due to the fact that the cost between all edges from the original graph $(u, v)$ and all edges between duplicates $(u', v')$ are set to infinity, they will never be part of $C^{\text{aux}}$. Thus, there are two types of edges contained in $C^{\text{aux}}$: $(u, v') \in E^{\text{aux}}$ representing an outgoing arc from node $u$ in $G_l$ and $(u', v) \in E^{\text{aux}}$ representing an ingoing arc to node $u$ in the original graph $G_l$. As every node $u$ has degree two in $C^{\text{aux}}$ and is connected with $u'$ there must be exactly one $v \neq u$ where $v'$ is connected with $u$. The same way there exists exactly one $w \neq u$ which is connected to $u'$. Consequently, every node has exactly one ingoing and exactly one outgoing arc in $C$. Since the undirected version of $C$ is exactly $C^{\text{aux}}$ after merging all vertices $v$ with $v'$, it can be concluded that $C$ is weakly connected. Since it was shown that every node has exactly one ingoing and exactly one outgoing arc and $C$ is weakly connected, consequently, $C$ has to be a Hamiltonian cycle in $G_l$. Moreover, if $C$ is a Hamiltonian cycle in $G_l$ we can construct the corresponding Hamiltonian cycle $C^{\text{aux}}$ in $G^{\text{aux}}$. Therefore, we have a bijection between all Hamiltonian cycles in $G_l$ and all Hamiltonian cycles in $\mathcal{C}$. The objectives of these Hamiltonian cycles is the same except a constant:

$$t_C = t_{C^{\text{aux}}} - (|V| + 2) \cdot M.$$

Therefore, if $C^{\text{aux}}$ is optimal, the corresponding $C$ also has to be optimal. Moreover, if $C$ is optimal, it follows that $C^{\text{aux}}$ has a minimum objective of all Hamiltonian cycles in $\mathcal{C}$. By construction all optimal Hamiltonian cycles of $G^{\text{aux}}$ have to be in $\mathcal{C}$ and therefore, $C^{\text{aux}}$ is optimal. $\square$

An optimal TSP solution on graph $G^{\text{aux}}$ will always connect node 0 to a visit of a pickup station $v \in V^{\text{pic}}$ since the costs for traveling from the depot 0 to a delivery station is infinity. Moreover, when a pickup station has been visited the next visit can only be performed at a delivery station $v \in V^{\text{del}}$ since costs for traveling between two pickup stations is also infinity. The same condition holds for traveling between two delivery stations. Traveling to the copy of the depot $0'$ can only be performed from a delivery station since the costs for traveling from a pickup station to the copy of the depot is infinity. Finally, the Hamiltonian path can be obtained by simply excluding the arc between 0 and $0'$ which is performed at no cost.

Technical Report AC-TR-15-004

Technical Report AC-TR-15-004

**Algorithm 1** LBBD for BBSS

```
1: repeat
2:    init: $r \leftarrow$ vector of $|L|$ empty routes, $cutsAdded \leftarrow$ false
3:    Solve MP to obtain subproblems
4:    for all $l \in L$ do
5:       $r_l \leftarrow$ solution of SP for vehicle $l$
6:       if $obj(r_l) > \hat{t}$ then
7:          $I \leftarrow V(r_l)$
8:          MP $\leftarrow$ MP $\cup \left(\sum_{v \in I} x_{vl} \leq |I| - 1 \quad \forall l \in L\right)$
9:          $cutsAdded \leftarrow$ true
10:      end if
11:   end for
12: until not($cutsAdded$)
13: return $r$
```

### 4.2.3. Iterated Decomposition Procedure and Cut Generation

Algorithm 1 shows an *LBBD* scheme utilizing cut generation by Benders infeasibility cuts. Variable $r$ denotes the current solution, i.e., the vector of $|L|$ routes, which are initially all empty. The function $obj(r_l)$ returns the objective value of a single subproblem solution, i.e., the actual routing costs when the TSP is solved to optimality.

The MP is solved in line (3) and the assignment of stations to vehicles is retrieved. We get our subproblems which are solved in the corresponding loop (4) for each vehicle separately. For every solution to a subproblem we utilize a solution cache. This means, that if a subproblem is feasible its corresponding Hamiltonian path and the routing costs are cached for later use. If the subproblem is infeasible it is not going to be cached because those subproblems result in a cut for the master problem. If we cannot find the subproblem in our solution cache, then a single subproblem is solved by Concorde (5) and added to the current solution $r$ as $r_l$. In the subproblem the routing costs are minimized and if this objective value is greater than the maximal time budget of the vehicles, we found an infeasible assignment (6). Let $I = \{r_l^1, r_l^2, \ldots, r_l^{\rho_l}\}$ be a set of station visits for which the minimal Hamiltonian path from 0 to $0'$ is greater than the time budget $\hat{t}$. Then, we can build infeasibility cuts of the form

$$\sum_{v \in I} x_{vl} \leq |I| - 1 \quad \forall l \in L. \tag{47}$$

These cuts are created for each vehicle $l \in L$ and added to the MP. They imply that the simultaneous assignment of the station visits in $I$ – and all supersets of $I$ – to any of the vehicles is prohibited in subsequent master problem instances.

To make this cut as strong as possible, we try to *minimize* the *infeasible* set $I$ of station visits, which is derived from all currently assigned stations (7) by Algorithm 2. Loop (1) iterates over all edges of a given Hamiltonian path $r_l = \{0, r_l^1, \ldots, r_l^{\rho_l}, 0'\}$ so that all possible options for minimizing the cutset are evaluated. We extract nodes $u$ and $v$ from the Hamiltonian path and refer the remaining set as $T$ (2). Two station

Technical Report AC-TR-15-004

**Algorithm 2** Minimize the cutset

```
init: $r_l \leftarrow$ Hamiltonian Path for vehicle $l$ in $G_l$, $\mathcal{I} \leftarrow \{I\}$, $\mathit{MinSize} \leftarrow |I|$
function $\mathit{minimizeCutSet}(r_l, \mathcal{I}, \mathit{MinSize})$
 1: for all $(u, v) \in r_l \mid u \notin \{0, 0'\}, v \notin \{0, 0'\}$ do
 2:   $T \leftarrow V(r_l) \setminus \{u, v\}$
 3:   $r'_l \leftarrow$ solution of SP for stations in $T$
 4:   if $\mathit{obj}(r'_l) > \hat{t}$ then
 5:     if $|T| = \mathit{MinSize}$ then
 6:       $\mathcal{I} \leftarrow \mathcal{I} \cup \{T\}$
 7:     else if $|T| < \mathit{MinSize}$ then
 8:       $\mathcal{I} \leftarrow \{T\}$
 9:       $\mathit{MinSize} \leftarrow |T|$
10:     end if
11:     $\mathcal{I} \leftarrow \mathit{minimizeCutSet}(r'_l, \mathcal{I}, \mathit{MinSize})$
12:     $\mathit{MinSize} \leftarrow |I'|, I' \in \mathcal{I}$
13:   end if
14: end for
15: return $\mathcal{I}$
```

visits have to be removed because the number of pickup and delivery station visits have to be equal in oder to obtain a feasible route. As edges can only exist between alternating station types it is ensured that only one pickup and one delivery station visit is removed. Here again, we utilize the proposed solution cache so that previously evaluated sets of station visits may not be evaluated multiple times. If the subproblem cannot be found in the solution cache, the subproblem of finding a Hamiltonian path for the reduced set of station visits in $T$ is solved (3) and the routing costs are checked for feasibility (4). If the set $T$ is infeasible we either found an additional cut (5) with equal size of station visits as the previous found cut(s) or we found a new cut containing less station visits than all previously found cuts (7). If the routing costs are feasible we did not find any new cut and do not have to explore this branch of the search tree further. If the routing costs have been infeasible, we recursively call the function *minimizeCutSet* (11) to check all subsets of $I$ which are candidates for a smaller cutset. At the end the set $I$ contains the smallest possible cutset(s) based on the initial one. It is also possible that $I$ contains more than one cut because multiple minimum cutsets may exist.

We can perform this algorithm because the subproblem is solved very efficiently by the Concorde TSP solver.

#### 4.2.4. Vehicle-spanning cuts

Due to the following observation we came up with the idea of also computing vehicle-spanning cuts instead of only utilizing cuts only for a single vehicle: Throughout the algorithm, a Benders infeasibility cut is added to the MP whenever an infeasible vehicle assignment is generated. In a new iteration of the master problem the MIP

Technical Report AC-TR-15-004

solver often tries to move station visits among different vehicles – because then these new assignments constitute different vehicles than in the cutset – although it may not be possible to come toward a feasible solution this way. The total (optimal) routing costs over all vehicles may be larger than the total amount of time budget provided by all available vehicles.

The idea is to solve the LP relaxation of the MIP formulation provided in Appendix B as it already provides reasonable lower bounds so that at least some of these vehicle-spanning cuts can be generated. Thus, we take the reduced set of station visits as solution from the MP but breakup the assignment to the vehicles and compute the minimal routing costs resulting from an optimal assignment. We therefore adjusted inequalities (27) from the master problem to the following equalities: $\sum_{l \in L} x_{vl} = 1 \; \forall v \in V$, and changed the objective function to minimize the total routing costs over all vehicles, i.e., min $\sum_{l \in L} \sum_{(u,v) \in A_0} y^l_{uv} \cdot t_{uv}$. If these routing costs are higher than the available time budget of all vehicles together, the set of station visits is not able to produce a feasible solution in any constellation of assignments. Let $h_l$ denote the minimal computed routing costs for the reduced set of station visits of vehicle $l \in L$, then we can add a vehicle-spanning cut iff

$$\sum_{l \in L} h_l > |L| \cdot \hat{t}. \tag{48}$$

Let $I$ denote the set of stations used in the currently considered assignment, i.e., $I = \{v \mid x_{vl} = 1, \forall v \in V, l \in L\}$. Assume, that inequality (48) holds for this assignment. Then formally, the cut is defined as follows:

$$\tag{49}$$

## 4.3. Branch-and-Check

As an alternative to the classical (logic-based) Benders decomposition method described in the previous section, we also consider a corresponding Branch-and-Check (BAC) approach. The term BAC has been originally proposed by Thorsteinsson in [47] and refers essentially to a classical branch-and-cut, which, however, is re-interpreted in terms of the Benders decomposition.

Instead of completely resolving the master problem after adding new Benders cuts in each iteration, BAC essentially solves the master problem only once and adds Benders cuts on the fly: BAC starts by solving the MIP model of the original master problem (possibly enhanced by a relaxation of the subproblem) in the usual branch-and-cut fashion. When an optimal solution is identified, the subproblem is solved by an auxiliary method and corresponding Benders cuts are derived as in the original Benders decomposition method. Now in BAC, however, these Benders cuts are dynamically added to the currently considered master problem within the branch-and-cut, effectively cutting off the current solution, and the branch-and-cut process is continued. Any obtained so far optimal solution to the master problem is “checked” in this way, and the master problem correspondingly augmented by Benders cuts until at some point the obtained solution turns out to be feasible and optimal also for the original

Technical Report AC-TR-15-004

problem; no further Benders cuts need then to be added, and the whole BAC can be terminated.

In our implementation of BAC we use again the MIP model proposed in Section 4.2.1. Remember that this model already includes the 0-arborescence problems as a relaxation of the subproblems. The subproblems are solved as described before via Concorde and the derivation of Benders Cuts follows exactly the already described way.

As an improvement to basic LBBD, the Benders subproblem is solved not only for so-far optimal master problem solutions, but for any encountered feasible master problem solution resulting in more cuts than the LBBD-based approach. The larger number of earlier added cuts turned out to be beneficial in the sense that the overall number of required branch-and-bound nodes and the overall runtime were reduced.

A particular advantage of this BAC is that it is in general able to yield upper as well as lower bounds and corresponding feasible solutions to the BBSS problem already early during the optimization. In contrast, classical Benders decomposition with infeasibility cuts is not directly able to provide lower bounds earlier than at the very end, as the master problem variables will not get overall feasible assignments before.

## 5. Variable Neighborhood Search

For comparison purposes and to further study the impact of the BBSS problem simplification by only considering full loads, we use here the VNS proposed by Rainer-Harbach et al. [40]. This VNS uses remove-station, insert-unbalanced-station, intra-route-2-opt, replace-station, intra-or-opt, 2-opt*-inter-route-exchange and intra-route 3-opt neighborhood structures for local improvement within an embedded Variable Neighborhood Descent (VND), and for shaking move-sequence, exchange-sequence, destroy-&-recreate, and remove-stations operations. The only modification we applied concerns the objective function, in which we set the weighting factors $\omega^{\text{work}}$ and $\omega^{\text{load}}$ for the additional terms to consider tour lengths and loading instructions to zero, in order to follow the same single goal of maximizing the balance gain as we do in our new approaches. Furthermore, as the balance gain is expressed in the VNS in terms of the number of bikes and in our case here in station visits, we scale the VNS's objective values accordingly by dividing them by the vehicle capacity.

## 6. Computational Results

We have done our computational tests on a rigorous benchmark suite derived from [40] with instances up to 70 and 120 stations for the multi-vehicle and the single-vehicle cases respectively. An instance is primarily characterized by the tuple of the number of stations, the number of vehicles, and the time limit $(|V|, |L|, \hat{t})$. All algorithms have been implemented in C++ and have been compiled with g++ 4.8.4. As MIP solver we used CPLEX 12.6.3 branch-and-cut with default parameters except limiting the number of used threads to one for a better comparability and restricting the maximum

Technical Report AC-TR-15-004

size of the branch-and-cut tree to 3GB as this is the amount of memory available to one cluster node in our configuration. All tests have been performed as single threads on an Intel Xeon E5540 2.53GHz Quad Core processor.

For our tests with multiple vehicles we use instances with 10, 20, 30, 40, 50, 60 or 70 stations, 2 or 3 vehicles, and a time budget of either 120, 240, or 480 minutes. For all instances we employed a maximum CPU time limit of 1 hour. For every combination $(|V|, |L|, \hat{t})$, we considered 30 different randomly generated instances which are available at `https://www.ac.tuwien.ac.at/files/resources/instances/bbss/benchs.tar.gz`. These instances have been derived from the real-world scenario at Citybike Wien, Vienna's major PBS in the following way. Citybike Wien gave us historic data sets from year 2011 when the system consisted of 92 renting stations—now the system has already 120 stations. We enlarged the dataset by 664 artificial stations placed among the city of Vienna. We have derived travel times $t_{uv}$ by an estimation of the real-world travel time and further including an estimation for the time needed to park at the station, i.e., some stations may be better reachable than others. For the existing stations we, of course, chose their original capacity $C_s$ whereas for the artificial ones we have chosen the capacities at random. The initial fill level of stations $p_s$ has been taken from a snapshot of 2011 for the existing stations. For the artificial stations we first chose for some of them initial fill levels at random according to a distribution we observed at the real stations so that geographically near stations have a similar fill level. For the other artificial stations we used an interpolation for determining their initial fill level. We set the target values $q_s$ in such a way that there are multiple full vehicle loads needed in order to bring the system in a balanced state. All tests inhere have been performed by considering a full vehicle load consisting of $Z = 20$ bikes. Of course, our approach works with arbitrary sizes of full vehicle loads, and how an instance for our new problem formulation is derived by the given data, is explained in the next paragraph. For every instance set we have chosen a corresponding subset of the 756 available stations where we first chose a station at random and then selected the $|S| + 1$ nearest stations. One station was randomly selected as the depot for the instance.

Following our new approach, we only consider the movement of full vehicle loads, i.e., $Z$ bikes from one station to another. Consequently, we derived each station's demand $f_s$ of full vehicle loads as defined in equations (8). Remember that delivery stations $S_{\text{del}}$ have negative demand values $f_s$, while pickup stations are given by positive values; stations with $|p_s - q_s| < Z$ do not need to be considered in our model and are therefore discarded. By above definition, it is ensured that we never move more than $|p_s - q_s|$ bikes to a station and thus never exceed a station's capacity.

In the following we analyze the results. Table 1 shows for every instance configuration results for the analyzed approaches: Logic-based Benders decomposition (LBBD) and its variant Branch-and-Check (BAC), compact MIP (see Section 2.3), and the variable neighborhood search (VNS) and the hop-indexed MIP from [40]. The column #best shows the number of instances where the particular approach achieved the best objective value among the 5 considered approaches. For BAC, LBBD and the compact MIP we show also the number of instances for which the corresponding approach returned a proven optimal solution #opt according to the new problem formulation.

Moreover, we also report the average objective value $\overline{obj}$ which corresponds to the station visits respectively twice the number of moved full vehicle loads. For BAC, LBBD and the compact MIP we take the optimal solution if available. If the optimal solution was not found we use the best integer solution found so far for BAC and the compact MIP. If also no integer solution was found we use 0 as the objective value for the corresponding instance. In the case of LBBD we take either the optimal solution or 0 because for this approach no bounds can be derived if the approach does not end up with a proven optimal solution. For VNS and miphop we use the original objective function as defined in [40] and shown in equation (4), only setting the weighting factors to $\omega^{\text{bal}} = 1$, $\omega^{\text{load}} = 0$, $\omega^{\text{work}} = 0$ accordingly, as already said we only consider balance optimization here. For comparing our new approaches with the VNS and miphop we use the following value:

$$\frac{\sum_{l \in L} \sum_{i=1}^{\rho_l} |y^i_{l,r^i_l}|}{Z}. \tag{50}$$

This value corresponds to twice the number of full vehicle loads moved among the stations and its average over a particular instance group is reported as $\overline{\text{obj}}$ in Table 1 for VNS and miphop.

In Section 6.1 we discuss the potential loss when considering only full vehicle loads. Then, in Section 6.2 we point out the advantages of our decomposition approaches and analyze the number of cuts and iterations of LBBD and its variant BAC and we analyze the approximation quality of the 0-arborescence in the master problem. Section 6.3 analyzes the single-vehicle case of our new problem formulation.

## 6.1. Analyzing the Impact of Considering Full Vehicle Loads Only

We first want to gain an approximate understanding of the loss in solution quality we obtain by moving from a previous *"detailed"* model, in which the loading and unloading of an arbitrary number of bikes is allowed, to our simplified model that considers only full vehicle loads.

For comparison in Table 1 we use a hop-indexed MIP model as well as one of the leading metaheuristic approaches, which is the VNS introduced in Section 5. Both approaches have been proposed in [40]. In the following we will concentrate on the VNS and BAC because these are the most competitive representatives of the two different problem formulations.

Remember that the VNS is not limited to full vehicle loads. It tries to find a tour together with best possible numbers of vehicles to load and unload at each stop. We just scaled the final overall balance gain, i.e., the sum of all loading instructions at each stop of every vehicle $\sum_{l \in L} \sum_{i=1}^{\rho_l} |y^i_{l,r^i_l}|$, by dividing it by the vehicle capacity $Z$, cf. equation (50), to make it comparable to the number of full vehicle loads by which we measure the balance gain with respect to the number of moved bikes in our new model.

Technical Report AC-TR-15-004

Table 1: This table provides a full comparison among the various discussed approaches.

| Instance set | | | BAC | | | | LBBD | | | | Compact | | | | VNS | | | miphop | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|S\|$ | $\|L\|$ | $\hat{t}$ | #best | #opt | $\overline{\text{obj}}$ | $\widetilde{\text{time}}\,[s]$ | #best | #opt | $\overline{\text{obj}}$ | $\widetilde{\text{time}}\,[s]$ | #best | #opt | $\overline{\text{obj}}$ | $\widetilde{\text{time}}\,[s]$ | #best | $\overline{\text{obj}}$ | $\widetilde{\text{time}}\,[s]$ | #best | $\overline{\text{obj}}$ | $\widetilde{\text{time}}\,[s]$ |
| 30 | 2 | 120 | **30** | 30 | 8.00 | 0.6 | **30** | 30 | 8.00 | 0.5 | **30** | 29 | 8.00 | 0.3 | 29 | 7.93 | 3600.0 | **30** | 8.00 | 10.4 |
| 30 | 2 | 240 | **26** | 27 | 18.80 | 23.2 | 23 | 27 | 17.00 | 13.8 | 24 | 25 | 18.67 | 19.2 | 14 | 17.87 | 3600.0 | 21 | 18.47 | 3600.0 |
| 30 | 2 | 480 | 11 | 30 | 26.47 | 1.0 | 11 | 30 | 26.47 | 0.5 | 2 | 30 | 25.33 | 0.2 | **23** | 28.87 | 3600.0 | 8 | 25.93 | 486.0 |
| 30 | 3 | 120 | **29** | 30 | 11.93 | 1.4 | **29** | 30 | 11.93 | 1.1 | **29** | 22 | 11.93 | 670.5 | 26 | 11.67 | 3600.0 | **29** | 11.93 | 35.7 |
| 30 | 3 | 240 | **22** | 24 | 25.27 | 35.5 | 18 | 24 | 20.33 | 28.0 | 17 | 30 | 24.80 | 4.1 | 20 | 25.00 | 3600.0 | 3 | 19.80 | 3600.0 |
| 30 | 3 | 480 | 11 | 30 | 26.47 | 3.6 | 11 | 30 | 26.47 | 0.4 | 2 | 30 | 25.33 | 0.4 | **23** | 28.87 | 3600.0 | 10 | 26.52 | 466.6 |
| 40 | 2 | 120 | **30** | 30 | 8.07 | 1.5 | **30** | 30 | 8.07 | 1.5 | **30** | 22 | 8.07 | 188.3 | 29 | 8.00 | 3600.0 | 29 | 8.00 | 33.4 |
| 40 | 2 | 240 | 23 | 25 | 19.60 | 44.8 | 23 | 26 | 17.27 | 82.4 | **24** | 20 | 19.67 | 43.6 | 11 | 18.53 | 3600.0 | 17 | 18.87 | 752.8 |
| 40 | 2 | 480 | 21 | 26 | 35.00 | 25.1 | 18 | 26 | 30.27 | 4.7 | 8 | 29 | 33.87 | 0.9 | **24** | 35.47 | 3600.0 | 0 | 7.38 | 3600.0 |
| 40 | 3 | 120 | **30** | 29 | 12.07 | 3.3 | 29 | 29 | 11.60 | 2.8 | **30** | 9 | 12.07 | 1146.8 | 25 | 11.73 | 3600.0 | 29 | 12.00 | 85.1 |
| 40 | 3 | 240 | **24** | 11 | 27.60 | 3600.0 | 8 | 8 | 7.93 | 3600.0 | 21 | 15 | 27.33 | 737.1 | 12 | 26.33 | 3600.0 | 5 | 20.07 | 3600.0 |
| 40 | 3 | 480 | 9 | 30 | 35.73 | 12.0 | 9 | 30 | 35.73 | 1.6 | 0 | 30 | 34.20 | 0.9 | **23** | 38.33 | 3600.0 | 5 | 23.67 | 3600.0 |
| 50 | 2 | 120 | 28 | 30 | 7.80 | 3.5 | 28 | 30 | 7.80 | 3.4 | **29** | 16 | 7.93 | 1108.8 | **29** | 7.93 | 3600.0 | **29** | 7.93 | 82.8 |
| 50 | 2 | 240 | **23** | 21 | 19.60 | 295.3 | 19 | 20 | 13.40 | 302.8 | 22 | 12 | 19.53 | 1383.6 | 21 | 19.27 | 3600.0 | 15 | 18.27 | 1518.9 |
| 50 | 2 | 480 | **22** | 5 | 38.40 | 3600.0 | 8 | 9 | 11.80 | 3600.0 | 21 | 23 | 38.40 | 114.2 | 17 | 37.87 | 3600.0 | 0 | 2.73 | 3600.0 |
| 50 | 3 | 120 | 28 | 29 | 11.67 | 7.4 | 28 | 29 | 11.40 | 10.8 | **29** | 9 | 11.87 | 1852.7 | 28 | 11.80 | 3600.0 | **29** | 11.87 | 198.5 |
| 50 | 3 | 240 | **23** | 9 | 28.13 | 3600.0 | 7 | 8 | 7.80 | 3600.0 | 21 | 7 | 27.93 | 1015.2 | 14 | 27.07 | 3600.0 | 4 | 14.33 | 3600.0 |
| 50 | 3 | 480 | 3 | 30 | 44.07 | 22.2 | 3 | 30 | 44.07 | 5.5 | 0 | 30 | 42.67 | 2.3 | **30** | 48.53 | 3600.0 | 0 | 4.33 | 3600.0 |
| 60 | 2 | 120 | **29** | 30 | 8.00 | 6.1 | 28 | 29 | 7.67 | 4.8 | **29** | 13 | 8.00 | 2362.9 | **29** | 8.00 | 3600.0 | 28 | 7.93 | 111.9 |
| 60 | 2 | 240 | 26 | 24 | 20.07 | 336.4 | 26 | 26 | 17.67 | 328.7 | **27** | 10 | 20.13 | 3589.9 | 14 | 19.00 | 3600.0 | 17 | 18.40 | 2410.3 |
| 60 | 2 | 480 | **23** | 2 | 40.00 | 3600.0 | 1 | 1 | 1.33 | 3600.0 | 20 | 17 | 39.80 | 604.0 | 12 | 38.33 | 3600.0 | 0 | 0.00 | 3600.0 |
| 60 | 3 | 120 | **29** | 28 | 11.93 | 17.1 | 28 | 28 | 11.20 | 16.6 | **29** | 5 | 11.93 | 3592.0 | 26 | 11.80 | 3600.0 | 28 | 11.60 | 269.2 |
| 60 | 3 | 240 | **24** | 11 | 28.67 | 3600.0 | 10 | 10 | 10.00 | 3600.0 | 22 | 3 | 28.53 | 1479.7 | 10 | 26.93 | 3600.0 | 0 | 5.93 | 3600.0 |
| 60 | 3 | 480 | 11 | 20 | 52.00 | 246.7 | 5 | 17 | 28.80 | 974.6 | 6 | 21 | 51.20 | 21.9 | **21** | 53.20 | 3600.0 | 0 | 0.00 | 3600.0 |
| 70 | 2 | 120 | **30** | 30 | 8.00 | 10.2 | **30** | 30 | 8.00 | 9.0 | **30** | 13 | 8.00 | 1255.9 | **30** | 8.00 | 3600.0 | **30** | 8.00 | 197.2 |
| 70 | 2 | 240 | **28** | 24 | 20.40 | 344.2 | 23 | 23 | 15.40 | 308.2 | **28** | 10 | 20.40 | 1826.4 | 17 | 19.60 | 3600.0 | 11 | 16.80 | 3326.7 |
| 70 | 2 | 480 | 20 | 3 | 41.17 | 3600.0 | 1 | 1 | 1.33 | 3600.0 | **23** | 10 | 41.33 | 928.6 | 9 | 39.53 | 3600.0 | 0 | 0.00 | 3600.0 |
| 70 | 3 | 120 | **30** | 30 | 12.00 | 27.9 | **30** | 30 | 12.00 | 25.6 | **30** | 2 | 12.00 | 3586.2 | 29 | 11.93 | 3600.0 | **30** | 12.00 | 439.8 |
| 70 | 3 | 240 | **27** | 16 | 29.60 | 2155.5 | 14 | 14 | 14.00 | 3600.0 | 24 | 1 | 29.40 | 2559.9 | 11 | 27.93 | 3600.0 | 0 | 1.00 | 3600.0 |
| 70 | 3 | 480 | **23** | 3 | 56.48 | 3600.0 | 1 | 3 | 5.47 | 3600.0 | 15 | 5 | 55.73 | 1103.5 | 12 | 55.13 | 3600.0 | 0 | 0.00 | 0.0 |
| | | **Total** | **693** | 667 | | | 529 | 658 | | | 622 | 498 | | | 618 | | | 407 | | |

Table 2: Additional aggregation provided by aggregating only over the number of nodes for logic-based Benders decomposition, its variant Branch-and-Check, and the compact mixed-integer linear programming model.

| **Instance set** | **BAC** | | | **LBBD** | | | **Compact** | | |
|---|---|---|---|---|---|---|---|---|---|
| $\|S\|$ | #best | #opt | $\widetilde{\text{time}}\,[s]$ | #best | #opt | $\widetilde{\text{time}}\,[s]$ | #best | #opt | $\widetilde{\text{time}}\,[s]$ |
| 10 | 180 | **180** | 0.1 | 180 | **180** | $< 0.1$ | 92 | 177 | $< 0.1$ |
| 20 | 180 | **180** | 0.4 | 179 | **180** | 0.2 | 110 | **180** | 0.1 |
| 30 | 178 | **171** | 2.2 | 170 | **171** | 0.9 | 135 | 166 | 0.8 |
| 40 | 176 | **151** | 12 | 149 | 149 | 4 | 133 | 125 | 36 |
| 50 | 174 | 124 | 89.8 | 124 | **126** | 62.2 | 151 | 97 | 558.9 |
| 60 | 171 | **115** | 343 | 111 | 111 | 352.4 | 155 | 69 | 1422.7 |
| 70 | 169 | **106** | 459.4 | 101 | 101 | 336.5 | 159 | 41 | 1673.6 |
| **Total** | 1228 | **1027** | | 1014 | 1018 | | 935 | 855 | |

When analyzing the results in Table 1 one can see that the average objective values obtained by BAC, which provided the best results for our new problem formulation, and the VNS correspond closely. Obviously, the simplification of considering only full vehicle loads has on these instances only a very minor impact. In fact, we could observe that also the solutions identified by the VNS also almost always transported only full vehicle loads from one station to another. Obviously, this only holds under our fundamental assumption that a complete balance with objective value zero, i.e., where all station demands are fulfilled, is not achievable within the limited working time – and also not necessary in practice. It can be observed that BAC yields more often the overall best solution—among the five different approaches—than the VNS (693 versus 618), and a Wilcoxon signed-rank test comparing BAC with the VNS on each instance set shows significant advantages with error probabilities of less than 5% for 12 of the 30 classes for BAC whereas the VNS has significant advantages on 5 of the 30 classes.

However, we have to note that the VNS often finds very fast high-quality solutions whereas, at least for the larger instances, the MIP-based approaches, namely LBBD, BAC, and the compact formulation often need a lot of time for finding an integer solution or even solve the problem instance to proven optimality. Smaller instances or mid-size instances, according to our test suite, are however, often solved to proven optimality within a small amount of time.

We conclude that the disadvantages of the simplified model are very well compensated by the much better solvability of the new approach.

## 6.2. Analyzing Logic-Based Benders Decomposition and the Compact Model

In Table 2 additional aggregation of the results is given by providing results for each individual number of stations. The best results are achieved by utilizing BAC. This

Technical Report AC-TR-15-004

Table 3: Comparing number of cuts, iterations, approximation quality and the time needed to solve master problems as well as subproblems.

| **Instance set** | | | **LBBD** | | | | | | **BAC** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\|S\|$ | $\|L\|$ | $\hat{t}$ | $\overline{\text{approx}}$ [%] | $\overline{\#\text{iter}}$ | $\overline{\#\text{cuts}}$ | $t^{\text{max}}_{\text{master}}$ [$s$] | $\overline{t_{\text{master}}}$ [$s$] | $t^{\text{max}}_{\text{sub}}$ [$s$] | $\overline{\text{approx}}$ [%] | $\overline{\#\text{calls}}$ | $\overline{\#\text{cuts}}$ | $t^{\text{max}}_{\text{sub}}$ [$s$] |
| 10 | 2 | 120 | 1.1 | 1.1 | 0.3 | 0.2 | 0.1 | $< 0.1$ | 1.8 | 1.4 | 0.3 | $< 0.1$ |
| 10 | 2 | 240 | 1.3 | 1.0 | 0.0 | 0.1 | $< 0.1$ | $< 0.1$ | 1.4 | 1.0 | 0.0 | $< 0.1$ |
| 10 | 2 | 480 | 1.3 | 1.0 | 0.0 | $< 0.1$ | $< 0.1$ | $< 0.1$ | 1.6 | 1.0 | 0.0 | $< 0.1$ |
| 10 | 3 | 120 | 0.5 | 1.1 | 0.4 | 0.3 | 0.1 | $< 0.1$ | 1.3 | 1.2 | 0.1 | $< 0.1$ |
| 10 | 3 | 240 | 1.0 | 1.0 | 0.0 | $< 0.1$ | $< 0.1$ | 0.1 | 1.3 | 1.1 | 0.0 | $< 0.1$ |
| 10 | 3 | 480 | 1.0 | 1.0 | 0.0 | $< 0.1$ | 0.1 | 0.1 | 1.4 | 1.0 | 0.0 | $< 0.1$ |
| 20 | 2 | 120 | 1.5 | 1.4 | 1.0 | 4.5 | 0.2 | $< 0.1$ | 2.1 | 1.8 | 0.7 | $< 0.1$ |
| 20 | 2 | 240 | 2.3 | 5.4 | 13.1 | 37.2 | 1.2 | 0.1 | 2.3 | 6.2 | 11.7 | 0.2 |
| 20 | 2 | 480 | 3.4 | 1.0 | 0.0 | 0.1 | $< 0.1$ | 0.1 | 2.5 | 1.1 | 0.0 | 0.6 |
| 20 | 3 | 120 | 1.4 | 1.8 | 4.3 | 15.3 | 0.8 | $< 0.1$ | 2.0 | 2.4 | 3.3 | $< 0.1$ |
| 20 | 3 | 240 | 2.0 | 1.2 | 0.5 | 0.5 | 0.2 | $< 0.1$ | 1.9 | 1.1 | 0.0 | $< 0.1$ |
| 20 | 3 | 480 | 2.4 | 1.0 | 0.0 | 0.1 | 0.1 | $< 0.1$ | 2.2 | 1.0 | 0.0 | $< 0.1$ |
| 30 | 2 | 120 | 1.2 | 1.4 | 0.9 | 1.4 | 0.4 | $< 0.1$ | 1.8 | 1.7 | 0.6 | 0.1 |
| 30 | 2 | 240 | 1.8 | 11.6 | 44.6 | 1925.7 | 37.5 | 0.9 | 1.8 | 21.2 | 63.9 | 0.6 |
| 30 | 2 | 480 | 4.8 | 1.0 | 0.0 | 0.6 | 0.2 | 0.1 | 4.2 | 1.1 | 0.0 | 0.1 |
| 30 | 3 | 120 | 1.3 | 2.2 | 4.5 | 39.6 | 2.1 | $< 0.1$ | 3.2 | 2.5 | 2.0 | $< 0.1$ |
| 30 | 3 | 240 | 1.9 | 4.6 | 21.5 | 3600.0 | 298.6 | 0.8 | 2.0 | 13.7 | 59.5 | 0.5 |
| 30 | 3 | 480 | 3.4 | 1.0 | 0.0 | 0.9 | 0.2 | 0.2 | 3.2 | 1.1 | 0.0 | 0.3 |
| 40 | 2 | 120 | 1.8 | 1.3 | 0.6 | 20.8 | 1.6 | $< 0.1$ | 3.0 | 2.2 | 1.1 | $< 0.1$ |
| 40 | 2 | 240 | 1.8 | 20.0 | 77.2 | 3407.2 | 81.0 | 0.6 | 1.7 | 47.8 | 159.1 | 0.5 |
| 40 | 2 | 480 | 3.3 | 4.3 | 11.0 | 3012.2 | 69.0 | 0.9 | 2.7 | 6.1 | 12.0 | 1.4 |
| 40 | 3 | 120 | 1.7 | 1.9 | 3.6 | 3600.0 | 122.2 | $< 0.1$ | 3.3 | 2.5 | 2.7 | 0.1 |
| 40 | 3 | 240 | 2.2 | 6.6 | 40.4 | 3600.0 | 1423.7 | 0.8 | 2.1 | 53.5 | 274.1 | 1.7 |
| 40 | 3 | 480 | 5.4 | 1.0 | 0.0 | 3.1 | 0.8 | 0.6 | 4.4 | 1.2 | 0.0 | 0.7 |
| 50 | 2 | 120 | 2.2 | 1.7 | 1.7 | 112.3 | 3.7 | $< 0.1$ | 2.4 | 2.1 | 1.1 | 0.1 |
| 50 | 2 | 240 | 2.5 | 27.1 | 115.2 | 3600.0 | 379.4 | 1.0 | 2.2 | 95.3 | 306.8 | 1.2 |
| 50 | 2 | 480 | 2.9 | 14.9 | 64.3 | 3600.0 | 704.0 | 1.1 | 2.7 | 97.2 | 406.0 | 3.2 |
| 50 | 3 | 120 | 1.9 | 2.6 | 7.8 | 1679.5 | 44.2 | $< 0.1$ | 2.8 | 3.5 | 7.2 | 0.1 |
| 50 | 3 | 240 | 2.6 | 6.8 | 44.8 | 3600.0 | 1171.4 | 0.3 | 2.4 | 49.9 | 267.8 | 0.7 |
| 50 | 3 | 480 | 5.4 | 1.0 | 0.0 | 5.2 | 2.8 | 0.5 | 4.0 | 2.1 | 0.1 | 0.3 |
| 60 | 2 | 120 | 1.9 | 2.1 | 2.6 | 988.7 | 25.3 | $< 0.1$ | 4.6 | 2.7 | 2.1 | 0.2 |
| 60 | 2 | 240 | 2.1 | 12.0 | 38.7 | 2510.5 | 80.8 | 0.3 | 2.2 | 53.8 | 176.0 | 0.9 |
| 60 | 2 | 480 | 3.0 | 11.4 | 49.4 | 3600.0 | 1530.3 | 1.9 | 2.8 | 55.8 | 195.0 | 2.0 |
| 60 | 3 | 120 | 2.2 | 2.0 | 4.9 | 3600.0 | 168.1 | $< 0.1$ | 3.4 | 4.6 | 9.3 | 0.2 |
| 60 | 3 | 240 | 2.6 | 6.7 | 40.0 | 3600.0 | 1036.4 | 0.5 | 2.1 | 36.0 | 173.9 | 0.8 |
| 60 | 3 | 480 | 3.5 | 1.6 | 4.4 | 3600.0 | 1006.9 | 0.5 | 2.9 | 18.4 | 77.4 | 4.0 |
| 70 | 2 | 120 | 1.6 | 1.3 | 0.6 | 17.7 | 6.8 | 0.1 | 5.3 | 2.0 | 0.9 | 0.2 |
| 70 | 2 | 240 | 1.6 | 2.6 | 5.9 | 3600.0 | 652.0 | 0.6 | 1.4 | 16.7 | 30.0 | 1.5 |
| 70 | 2 | 480 | 3.1 | 11.5 | 48.8 | 3600.0 | 1758.4 | 1.1 | 2.2 | 60.1 | 188.3 | 2.3 |
| 70 | 3 | 120 | 1.5 | 1.7 | 2.6 | 33.2 | 13.8 | $< 0.1$ | 4.2 | 4.7 | 8.6 | 0.4 |
| 70 | 3 | 240 | 2.0 | 4.2 | 22.7 | 3600.0 | 1053.2 | 0.5 | 1.8 | 22.6 | 97.9 | 1.3 |
| 70 | 3 | 480 | 4.3 | 1.0 | 8.3 | 3600.0 | 2429.6 | 1.7 | 2.7 | 15.7 | 60.4 | 1.6 |
| | | **Average** | 2.3 | 4.5 | 16.3 | | | | 2.6 | 17.1 | 61.9 | |

Technical Report AC-TR-15-004

Technical Report AC-TR-15-004

is also the approach which yields the highest number of proven optimal solutions among the analyzed dataset. As expected the median computation times are better by applying LBBD than solving the full compact model at once. This further emphasizes the importance of the decomposition approach we are introducing in this paper.

In Table 3 we measure various statistics about LBBD and BAC. Column $\overline{\text{approx}}$ shows the average approximation quality of the 0-arborescence as percentage value which is computed by $(t^{\text{est}} - t^{\text{opt}})/t^{\text{opt}}$, where $t^{\text{est}}$ are the estimated routing costs by the 0-arborescence and $t^{\text{opt}}$ are the optimal routing costs as obtained by the optimal solution for a subproblem. The average over all computed routes is taken. Column $\overline{\#\text{iter}}$ shows the average number of iterations performed per instance set whereas $\overline{\#\text{cuts}}$ shows the average number of generated cuts. For BAC we use a different nomenclature, namely $\overline{\#\text{calls}}$ as for this variant of LBBD we do not have iterations but we measure the calls to the LazyConstraintCallback where we generate the Benders infeasibility cuts. Moreover, columns $t^{\max}_{\text{master}}$ and $\overline{t_{\text{master}}}$ state the maximum time used to solve the master problem and the average time respectively. Column $t^{\max}_{\text{sub}}$ shows the maximum time which was needed to solve a single subproblem per instance set.

A result of this comparison is that subproblems are relatively easy to solve compared to the master problems. As seen in Table 3 the most difficult subproblem needed only 4 seconds to be solved whereas some of the the master problem instances could not be solved to optimality within 1 hour. What can also be seen is that the approximation via the 0-arborescence is tight. In many cases we have only 1 to 2 percent deviation from the optimal routing costs. Furthermore, as expected, subproblems for instances with low time budget, i.e., 2 hours, are very easy to solve and most often even do not need 0.1 seconds to be solved. BAC generates much more cuts as LBBD evaluates subproblems only after optimal solutions to the master problem have been found whereas BAC evaluates subproblems each time a feasible integer solution has been found. Although, BAC generates more cuts than LBBD, the quantity of cuts is no guarantee for success at all, but the quality of the cuts can improve solvability of the problem instance. A phenomenon we could observe is that often instances with 4 hours time limit are more difficult to solve and also produce more cuts/iterations. This is because the objective of the problem is only to maximize the number of station visits. Thus, multiple optimal solutions may exist, even if stations are near to another. As already said, as the approximation quality of the 0-arborescence is relatively tight one of the optimal solutions to an instance with a higher time limit, i.e., 8 hours can be obtained very fast. On the other hand there are instances with 4 hours time limit that are hard to solve because there may not be as many optimal solutions as for instances with 8 hours time limit, even for the smaller instances. What has been observed by the authors and which was the reason to introduce vehicle-spanning cuts (see also Section 4.2.4) is that the approach often needs many iterations to prove that there does not exist a solution with a given number of station visits but there could be many assignments with 2 visits less that are optimal.

Technical Report AC-TR-15-004

Table 4: Computational results for the single-vehicle case

| Instance set | | Compact | | | | | |
|---|---|---|---|---|---|---|---|
| $\|V\|$ | $\hat{t}$ | #opt | $\overline{\text{obj}}$ | $\overline{\text{LB}}$ | $\overline{\text{UB}}$ | $\overline{\text{gap}}\,[\%]$ | $\widetilde{\text{time}}\,[s]$ |
| 40 | 120 | 30 | 4.07 | 4.07 | 4.07 | 0.00 | 0.1 |
| 40 | 240 | 30 | 10.13 | 10.13 | 10.13 | 0.00 | 0.2 |
| 40 | 480 | 29 | 20.93 | 20.93 | 20.98 | 0.24 | 0.4 |
| 50 | 120 | 30 | 4.00 | 4.00 | 4.00 | 0.00 | 0.1 |
| 50 | 240 | 28 | 10.33 | 10.33 | 10.43 | 1.11 | 0.4 |
| 50 | 480 | 29 | 21.53 | 21.53 | 21.59 | 0.30 | 1.9 |
| 60 | 120 | 30 | 4.07 | 4.07 | 4.07 | 0.00 | 0.2 |
| 60 | 240 | 23 | 10.33 | 10.33 | 10.66 | 3.29 | 1.6 |
| 60 | 480 | 27 | 22.00 | 22.00 | 22.16 | 0.77 | 7.1 |
| 70 | 120 | 30 | 4.00 | 4.00 | 4.00 | 0.00 | 0.3 |
| 70 | 240 | 22 | 10.53 | 10.53 | 10.91 | 3.76 | 6.8 |
| 70 | 480 | 21 | 22.47 | 22.47 | 22.90 | 2.03 | 15.2 |
| 90 | 120 | 29 | 4.00 | 4.00 | 4.05 | 1.13 | 0.5 |
| 90 | 240 | 18 | 10.53 | 10.53 | 11.12 | 5.84 | 98.3 |
| 90 | 480 | 17 | 22.73 | 22.73 | 23.34 | 3.10 | 23.1 |
| 120 | 120 | 29 | 4.00 | 4.00 | 4.05 | 1.13 | 1.3 |
| 120 | 240 | 13 | 10.60 | 10.60 | 11.51 | 9.06 | 3120.2 |
| 120 | 480 | 19 | 23.27 | 23.27 | 23.88 | 2.76 | 369.6 |
| **Average** | | | 12.20 | 12.20 | 12.44 | 1.92 | |

## 6.3. Single-Vehicle Case

We have also performed computational tests on the single-vehicle case of the problem which we have solved by the compact MIP model provided in Appendix A. This is also of practical importance for cases where the whole service area is more statically partitioned into districts and individual drivers/vehicles are then solely responsible for dedicated districts. Computational results are shown in Table 4. As the problem becomes simpler when reducing the number of vehicles we have also provided results for instances with 90 and 120 stations. For the most difficult instances with 120 stations and a shift time of 4 hours for the driver we have been able to solve 13 out of 30 instances to proven optimality. For the remaining instances we have been able to provide results with an average optimality gap of about 5%.

# 7. Conclusions and Future Work

We have introduced and investigated a new problem formulation for BBSS derived from practical considerations as they appear, e.g., at Citybike Wien, which is computationally substantially simpler to solve than previous BBSS formulations. The key observation is that an economic maintenance of a PBS rarely allows to bring all stations into perfect balance w.r.t. precisely specified target fill levels. In practice, usually "more rebalancing work actually exists than can be typically achieved" with the given number of vehicles and limited working time of the drivers. Therefore, vehicles almost always move full vehicle loads among the stations. Moving just very few bikes from one station to another is basically meaningless in practice.

While previous BBSS models always considered a rather fine-grained planning allowing the movement of arbitrary numbers of bikes, we restrict our rebalancing tours to the movement of full vehicle loads from the beginning. This restriction yields substantial simplifications in the overall model, and consequently, the model can be solved computationally significantly easier.

For solving this new model, we developed a compact MIP model, an LBBD approach, and a BAC variant of the latter. The LBBD was inspired by the cluster-first route-second method because the problem can naturally be split in an assignment part and a routing part. The integral subproblems turned out to essentially correspond to asymmetric TSPs, which we solve by the state-of-the-art TSP solver Concorde. From these, Bender's infeasibility cuts are derived and iteratively added to the assignment master problem. In the BAC, we modified the LBBD approach by solving the master problem only once and solving corresponding subproblems for any encountered feasible solution. This modification turned out to further improve the performance in many cases.

Experimental comparisons with a state-of-the-art VNS for a previous fine-grained BBSS model have shown that our new problem formulation has only a minor impact on the achievable solution quality. In fact, the advantages of the easier solvability clearly outweigh the theoretical restrictions introduced by allowing only full vehicle loads. A Wilcoxon signed-rank test showed that BAC compared to VNS has significant advantages with an error probability of less than 5% for 12 of the 30 instance sets.

With our LBBD we could solve instances up to 70 stations to proven optimality, which is a substantial step forward in comparison to previous work with exact approaches. For the single-vehicle problem, we have solved even larger instances up to 120 stations.

Clearly, the proposed compact MIP, LBBD, and BAC are not the only meaningful methods to approach the new simplified BBSS problem formulation exactly. State-of-the-art branch-and-cut solvers for diverse vehicle routing problem variants based on subtour-elimination constraints can likely be adapted and are then presumably strong if not superior competitors. Also column generation approaches based on some set covering formulation appear meaningful.

But also in a purely heuristic context for addressing larger problem instances with possibly thousands of stations, the simplified modeling approach appears very meaningful and opens diverse existing methods for vehicle routing problem variants to be adapted with moderate effort.

Technical Report AC-TR-15-004

## Acknowledgments

This work is supported by the Austrian Research Promotion Agency (FFG) under contract 849028. We thank Bin Hu, Matthias Prandtstetter, Andrea Rendl, Christian Rudloff, and Markus Straub from the Austrian Institute of Technology (AIT) for the collaboration in this project, constructive comments and for providing the data used in our test instances. In addition, we thank Citybike Wien for providing information about practical aspects of their bike sharing system.

## References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *TSP Cuts Which Do Not Conform to the Template Paradigm* pp. 261–303, Springer Berlin Heidelberg 2001.

[2] R. Aringhieri, M. Bruglieri, F. Malucelli, and M. Nonato, Metaheuristics for a Vehicle Routing Problem on Bipartite Graphs with distance constraints, 6th Metaheuristics Int Conference, Vienna, Austria, 2005, pp. 77–82.

[3] E. Balas, The prize collecting traveling salesman problem, Networks 19 (1989), 621–636.

[4] A. Baltz and A. Srivastav, Approximation algorithms for the euclidean bipartite TSP, Oper Res Lett 33 (2005), 403–410.

[5] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numer Mathematik 4 (1962), 238–252.

[6] D. Chemla, F. Meunier, and R.W. Calvo, Bike sharing systems: Solving the static rebalancing problem, Discr Optim 10 (2013), 120–146.

[7] G. Codato and M. Fischetti, Combinatorial Benders' cuts for mixed-integer linear programming, Oper Res 54 (2006), 756–766.

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

[9] W. Cook, Concorde TSP Solver, webpage. `http://www.math.uwaterloo.ca/tsp/concorde/` [Online; accessed 06-May-2015].

[10] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, The bike sharing rebalancing problem: Mathematical formulations and benchmark instances, Omega 45 (2014), 7–19.

[11] L. Di Gaspero, A. Rendl, and T. Urli, Constraint-based approaches for balancing bike sharing systems, Principles Practice Constraint Program, Vol. 8124 of *LNCS*, Springer, 2013, pp. 758–773.

Technical Report AC-TR-15-004

[12] L. Di Gaspero, A. Rendl, and T. Urli, A hybrid ACO+CP for balancing bicycle sharing systems, Hybrid Metaheuristics, Vol. 7919 of *LNCS*, Springer, 2013, pp. 198–212.

[13] G. Erdoğan, G. Laporte, and R.W. Calvo, The static bicycle relocation problem with demand intervals, Eur J Oper Res 238 (2014), 451–457.

[14] M.L. Fisher and R. Jaikumar, A generalized assignment heuristic for vehicle routing, Networks 11 (1981), 109–124.

[15] I.A. Forma, T. Raviv, and M. Tzur, A 3-step math heuristic for the static repositioning problem in bike-sharing systems, Transportation Res Part B: Methodological 71 (2015), 230–247.

[16] A. Fránk, E. Triesch, B. Korte, and J. Vygen, On the bipartite travelling salesman problem, Technical report 98866-OR, Research Institute for Discrete Mathematics, 1998.

[17] C. Fricker and N. Gast, Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity, EURO J Transportation Logist (2014), 1–31.

[18] M. Gendreau, J. Nossack, and E. Pesch, Mathematical formulations for a 1-full-truckload pickup-and-delivery problem, Eur J Oper Res 242 (2015), 1008–1016.

[19] Y. Han, E. Côme, and L. Oukhellou, Towards Bicycle Demand Prediction of Large-Scale Bicycle Sharing System, Transportation Res Board 93rd Ann Meeting, 2014, pp. 1–17.

[20] I. Harjunkoski and I.E. Grossmann, Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods, Comput & Chemical Eng 26 (2002), 1533–1552.

[21] H. Hernández-Pérez and J.J. Salazar-González, A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, Discr Appl Math 145 (2004), 126–139.

[22] H. Hernández-Pérez, I. Rodríguez-Martín, and J.J. Salazar-González, A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem, Comput & Oper Res 36 (2009), 1639–1645.

[23] H. Hernández-Pérez and J.J. Salazar-González, Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, Transportation Sci 38 (2004), 245–255.

[24] H. Hernández-Pérez and J.J. Salazar-González, The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms, Networks 50 (2007), 258–272.

Technical Report AC-TR-15-004

[25] S.C. Ho and W. Szeto, Solving a static repositioning problem in bike-sharing systems using iterated tabu search, Transportation Res Part E: Logist Transportation Review 69 (2014), 180–198.

[26] J. Hooker, Planning and scheduling by logic-based Benders decomposition, Oper Res 55 (2007), 588–602.

[27] J.N. Hooker and G. Ottosson, Logic-based Benders decomposition, Math Program 96 (2003), 33–60.

[28] R. Jonker and T. Volgenant, Transforming asymmetric into symmetric traveling salesman problems, Oper Res Lett 2 (1983), 161–163.

[29] R. Jonker and T. Volgenant, Transforming asymmetric into symmetric traveling salesman problems: erratum, Oper Res Lett 5 (1986), 215–216.

[30] C. Kloimüllner, P. Papazek, B. Hu, and G.R. Raidl, Balancing bicycle sharing systems: An approach for the dynamic case, EvoCOP, Vol. 8600 of *LNCS*, Springer, 2014, pp. 73–84.

[31] M.C. Lai, H. Sohn, and D. Bricker, A hybrid Benders/genetic algorithm for vehicle routing and scheduling problem, Int J Indust Eng: Theory, Appl Practice 19 (2012).

[32] G. Laporte, F. Meunier, and R.W. Calvo, Shared mobility systems, 4OR 13 (2015), 341–360.

[33] J.R. Lin and T.H. Yang, Strategic design of public bicycle sharing systems with service level constraints, Transportation Res Part E: Logist Transportation Review 47 (2011), 284–294.

[34] C.E. Miller, A.W. Tucker, and R.A. Zemlin, Integer programming formulation of traveling salesman problems, J ACM 7 (1960), 326–329.

[35] P. Papazek, C. Kloimüllner, B. Hu, and G.R. Raidl, Balancing bicycle sharing systems: An analysis of path relinking and recombination within a GRASP hybrid, PPSN, Vol. 8672 of *LNCS*, Springer, 2014, pp. 792–801.

[36] P. Papazek, G.R. Raidl, M. Rainer-Harbach, and B. Hu, A PILOT/VND/GRASP hybrid for the static balancing of public bicycle sharing systems, Proc 14th EUROCAST, Vol. 8111 of *LNCS*, Springer, 2013, pp. 372–379.

[37] J. Pfrommer, J. Warrington, G. Schildbach, and M. Morari, Dynamic vehicle redistribution and online price incentives in shared mobility systems, Intelligent Transportation Syst, IEEE Trans 15 (2014), 1567–1578.

[38] G.R. Raidl, B. Hu, M. Rainer-Harbach, and P. Papazek, Balancing bicycle sharing systems: Improving a VNS by efficiently determining optimal loading operations, Hybrid Metaheuristics, Vol. 7919 of *LNCS*, Springer, 2013, pp. 130–143.

Technical Report AC-TR-15-004

[39] M. Rainer-Harbach, P. Papazek, B. Hu, and G.R. Raidl, Balancing bicycle sharing systems: A variable neighborhood search approach, EvoCOP, Vol. 7832 of *LNCS*, Springer, 2013, pp. 121–132.

[40] M. Rainer-Harbach, P. Papazek, B. Hu, G.R. Raidl, and C. Kloimüllner, PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems, J Global Optim 63 (2015), 597–629.

[41] T. Raviv, M. Tzur, and I.A. Forma, Static repositioning in a bike-sharing system: models and solution approaches, EURO J Transportation Logist 2 (2013), 187–229.

[42] C. Rudloff and B. Lackner, Modeling demand for bikesharing systems, Transportation Res Record: J Transportation Res Board 2430 (2014), 1–11.

[43] J.J. Salazar-González and B. Santos-Hernández, The split-demand one-commodity pickup-and-delivery travelling salesman problem, Transportation Res Part B: Methodological 75 (2015), 58–73.

[44] J. Schuijbroek, R. Hampshire, and W.J. van Hoeve, Inventory rebalancing and vehicle routing in bike sharing systems, Eur J Oper Res 257 (2017), 992–1004.

[45] A. Shurbevski, H. Nagamochi, and Y. Karuno, Approximating the bipartite TSP and its biased generalization, Algorithms Computation, Vol. 8344 of *LNCS*, Springer, 2014, pp. 56–67.

[46] A. Srivastav, H. Schroeter, and C. Michel, Approximation algorithms for pick-and-place robots, Ann Oper Res 107 (2001), 321–338.

[47] E.S. Thorsteinsson, Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming, Principles Practice Constraint Programming—CP 2001, Vol. 2239 of *LNCS*, Springer, 2001, pp. 16–30.

[48] C.K. Ting and X.L. Liao, The selective pickup and delivery problem: Formulation and a memetic algorithm, Int J Production Economics 141 (2013), 199–211.

[49] P. Vogel, B.A. Neumann Saavedra, and D.C. Mattfeld, A hybrid metaheuristic to solve the resource allocation problem in bike sharing systems, Hybrid Metaheuristics, Vol. 8457 of *LNCS*, Springer, 2014, pp. 16–29.

[50] S. Voß, A. Fink, and C. Duin, Looking ahead with the PILOT method, Ann Oper Res 136 (2005), 285–302.

Technical Report AC-TR-15-004

## A. Single-vehicle MIP model

The following MIP formulation models the problem, if only a single vehicle is applied to BBSS.

$$
\begin{aligned}
\max \quad & \sum_{v\in V} x_v && && (51)\\
\text{s.t.} \quad & x_v \le 1 && \forall v \in V && (52)\\
& \sum_{v\in V_{\text{pic}}} x_v = \sum_{v\in V_{\text{del}}} x_v && && (53)\\
& x_{(s,i)} \ge x_{(s,i+1)} && \forall s \in S, i = 1,\dots,f_s - 1 && (54)\\
& \sum_{v\in V_{\text{pic}}} y_{0v} = 1 && && (55)\\
& \sum_{v\in V_{\text{del}}} y_{v0'} = 1 && && (56)\\
& \sum_{(u,v)\in A_0} y_{uv} = x_u && \forall u \in V && (57)\\
& \sum_{(u,v)\in A_0} y_{uv} = x_v && \forall v \in V && (58)\\
& \sum_{(u,v)\in A_0} y_{uv} = \sum_{(v,u)\in A_0} y_{vu} && \forall v \in V && (59)\\
& a_u - a_v + |V| \cdot y_{uv} \le |V| - 1 && \forall (u,v) \in A && (60)\\
& \sum_{(u,v)\in A_0} y_{uv} \cdot t_{uv} \le \hat{t} && && (61)\\
& x_v \in \{0,1\} && \forall v \in V && (62)\\
& y_{uv} \in \{0,1\} && \forall (u,v) \in A_0 && (63)\\
& 1 \le a_v \le |V| && \forall v \in V && (64)
\end{aligned}
$$

## B. Routing MIP model

This MIP formulation can be used to obtain minimal routing costs for a predefined set of station visits.

$$
\begin{aligned}
\min \quad & \sum_{l\in L} \sum_{(u,v)\in A_0} y^l_{uv} \cdot t_{uv} && && (65)\\
\text{s.t.} \quad & \sum_{l\in L} x_{vl} = 1 && \forall v \in V && (66)\\
& \sum_{v\in V_{\text{pic}}} x_{vl} = \sum_{v\in V_{\text{del}}} x_{vl} && \forall l \in L && (67)\\
& \sum_{l'\in L} x_{(s,i)l'} \ge x_{(s,i+1)l} && \forall s \in S, l \in L, i = 1,\dots,f_s - 1 && (68)
\end{aligned}
$$

Technical Report AC-TR-15-004

$$\sum_{v \in V_{\text{pic}}} y^l_{0v} = 1 \qquad \forall l \in L \tag{69}$$

$$\sum_{v \in V_{\text{del}}} y^l_{v0'} = 1 \qquad \forall l \in L \tag{70}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{ul} \qquad \forall l \in L, u \in V \tag{71}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{vl} \qquad \forall l \in L, v \in V \tag{72}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = \sum_{(v,u) \in A_0} y^l_{vu} \qquad \forall l \in L, v \in V \tag{73}$$

$$a_u - a_v + |V| \cdot y^l_{uv} \leq |V| - 1 \qquad \forall l \in L, (u,v) \in A \tag{74}$$

$$x_{vl} \in \{0,1\} \qquad \forall l \in L, v \in V \tag{75}$$

$$y^l_{uv} \in \{0,1\} \qquad \forall l \in L, (u,v) \in A_0 \tag{76}$$

$$1 \leq a_v \leq |V| \qquad \forall v \in V \tag{77}$$

Technical Report AC-TR-15-004