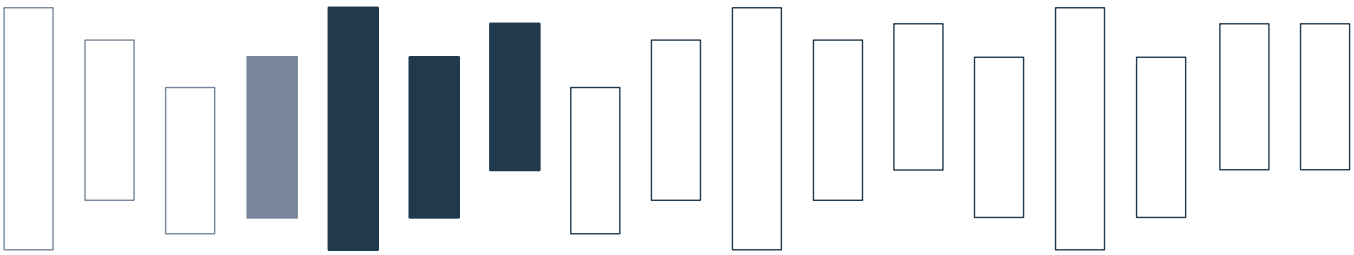
Technical Report AC-TR-15-004

November 2015

Full-Load Route Planning for Balancing Bike Sharing Systems by Logic-Based Benders Decomposition and Branch-and-Check

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

Full-Load Route Planning for Balancing Bike Sharing Systems by Logic-Based Benders Decomposition and Branch-and-Check

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, 2015

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 branch-and-check approach 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

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. PBSs can help in solving the last mile problem arising in public transport, and they encourage the population to do more sports by riding bikes. They are a contribution to *smart cities* as they reduce motorized traffic and are more ecofriendly [9]. Moreover, bikes also need less parking space than cars which can help to utilize limited space more efficiently.

A PBS should not be confused with classical bike rental as both have different use cases, client bases and revenue models. The major differences are that in PBS short-term usage is promoted whereas in bike rental longer rental times are not unusual, PBSs are distributed over a larger area, whereas bike rentals are more stationary with bikes usually to be returned at the same place where they have been rent [38].

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 [28] 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 [15, 32]. 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* [35]. 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 modelling 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 branch-and-check (BAC). Moreover, we compare with previously proposed 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 4 the compact MIP model is introduced, whereas Section 5 describes the LBBD and Section 5.4 the related BAC. Computational results are shown in Section 7, and finally, we conclude

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

in Section 8.

2. Problem Formulation

We first summarize aspects of existing problem formulations for *Balancing Bike Sharing Systems* (BBSSs) 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.

For the static case, a significant variety of different optimization goals has been considered in the literature, e.g., minimizing the traveling costs [4, 8] where balancing is modeled as a hard constraint, or minimizing the total number of expected shortages [36].

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. Obviously, it depends on the customer demand how the PBS operator fixes the target values or target intervals for rebalancing as it is necessary to cover the future demand of the customers. Thus, a sophisticated demand prognosis is necessary to estimate well-suited target values. Rudloff and Lackner [37] 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. [16] 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 [36].

The *dynamic case* also considers user interactions during rebalancing. Only few works, however, exist in this direction. In [26] the user interactions and the demands are retrieved from historical data and implemented by a probabilistic model obtained by Rudloff and Lackner [37], 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. [6] 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 the pickup and delivery of full vehicle loads clearly dominate 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 (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. Furthermore, many stations ideally require more than one, sometimes even several full vehicle loads to be 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, the time differences for loading or unloading less or more bikes, however, 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 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 ($f_s \geq 1$) or picked up ($f_s \leq -1$) 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 we can safely assume 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 $V_{\text{pic}} = \{s_i \mid s \in S \wedge f_s \leq -1,\ i = 1, \ldots, |f_s|\}$ be a set of nodes representing up to $|f_s|$ visits at each pickup station, and let $V_{\text{del}} = \{s_i \mid s \in S \wedge f_s \geq 1,\ i = 1, \ldots, f_s\}$ denote the respective potential visits at the delivery stations. $V = V_{\text{pic}} \cup V_{\text{del}}$ then refers to the joined set of all potential visits, and the arc set of graph G is given by $A = \{(u, v), (v, u) \mid u \in V_{\text{pic}}, v \in V_{\text{del}}\}$.

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., $A_0 = A \cup \{(0, v) \mid v \in V_{\text{pic}}\} \cup \{(v, 0') \mid v \in V_{\text{del}}\}$, 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 location u to location 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 $|L|$ Hamiltonian paths starting at 0 and ending at $0'$, or in other words a set of $|L|$ simple disjoint paths in G_0 from node 0 to node $0'$. 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 ρ_l being the number of visits. Due to the bipartite structure of G_0, as long as the path is not empty ($\rho_l > 0$) each odd stop must be at a pickup station, i.e., $r_l^1, r_l^3, \ldots, r_l^{\rho_l - 1} \in V_{\text{pic}}$, while each even stop takes place at a delivery station, i.e., $r_l^2, r_l^4, \ldots, r_l^{\rho_l} \in V_{\text{pic}}$, and ρ_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 [35] 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

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

and is the primary objective of previous work such as [10, 11, 22, 35, 36].

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 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.

3.1. MIP Approaches

Chemla et al. [4] 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. [36] 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. [39] 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. Concerning the considered problem variant, there are also several other differences to our approach: For each station, Schuijbroek et al. assume a target demand interval as given and it is considered a hard constraint to bring the fill level of each station within this interval. Shift times are not considered, and it is the main objective to minimize the overall makespan for the whole rebalancing. As travel-distance matrix they use Euclidean distances which does not reflect reality well. For deriving clusters of stations they use a maximum spanning star (MAXSPS) model to approximate the routing costs within each cluster whereas we will utilize an arborescence approximation rooted at the depot for the traveling salesman problem (TSP). They show computational results on the systems of *Capital Bikeshare* located in Washington, DC with 135 stations and *Hubway* in Boston, MA with 60 stations. They compare four different approaches of their own: a compact MIP model, a clustered MIP, a clustered MIP with cuts and a constraint programming approach. However, on the Boston instances only 10 stations lie originally outside of the target service level interval and in the Washington, DC instances 11 respectively 25 stations. The instances for Capital Bikeshare have been too complex for retrieving any feasible solution through the compact MIP model.

Erdoğan et al. [12] define demand intervals for each station which have to be satisfied similarly as it is also done by Schuijbroek et al. 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. [34] 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. [31] have developed a pilot heuristic [44] which improved the GCH from [34] significantly, a greedy randomized adaptive search procedure (GRASP) upon both construction heuristics performing very well on instances with a high number of rental stations. Raidl et al. [33] 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. [35] 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. [30] investigated diverse path relinking extensions for GRASP.

The dynamic case was considered by Kloimüllner et al. [26], 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 [35]. The VNS was able to solve instances with up to 90 stations reasonably well.

Di Gaspero et al. further describe a constraint programming approach [10] and a hybridization of it with ant colony optimization [11]. They tested on the same benchmark set as Rainer-Harbach et al. [35]. 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. [43] 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. [14] 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 [36]. 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. [1] 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 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.

Related to our problem formulation also is the one-commodity pickup and delivery traveling salesman problem (1-PDTSP) described by Hernández-Pérez et al. [18, 19, 20, 21], and the selective pickup and delivery problem (SPDP) studied by Ting et al. [42]. 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 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 [2], 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 5, we obtain as subproblem independent Hamiltonian path problems for the individual vehicles. These problems can be modeled classical asymmetric TSPs (ATSP) on bipartite graphs. Concerning this special TSP variant, not much specific work exists. Shurbevski et al. [40] proposed several approximation algorithms, which are, however, more of theoretical interest. We will apply the well-known *Concorde* TSP solver [7] to tackle these subproblems, not further exploiting the underlying bipartite graph structure.

4. Compact MIP Formulation

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* $y^l_{uv} \in \{0,1\}$ to describe the tour for each vehicle. Subtours are eliminated via Miller-Tucker-Zemlin inequalities [29] utilizing further continuous variables a_i for the nodes $i \in V$.

$$\max \quad \sum_{l \in L} \sum_{v \in V} x_{vl} \tag{4}$$

$$\text{s.t.} \quad \sum_{l \in L} x_{vl} \leq 1 \qquad \forall v \in V \tag{5}$$

$$\sum_{v \in V_{\text{pic}}} x_{vl} = \sum_{v \in V_{\text{del}}} x_{vl} \qquad \forall l \in L \tag{6}$$

$$x_{s_i l} \geq x_{s_{i+1} l} \qquad \forall s \in S,\ l \in L,\ i = 1, \ldots, f_s - 1 \tag{7}$$

$$\sum_{v \in V_{\text{pic}}} y^l_{0v} = 1 \qquad \forall l \in L \tag{8}$$

$$\sum_{v \in V_{\text{del}}} y^l_{v0'} = 1 \qquad \forall l \in L \tag{9}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{ul} \qquad \forall l \in L,\ u \in V \tag{10}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{vl} \qquad \forall l \in L,\ v \in V \tag{11}$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = \sum_{(v,u) \in A_0} y^l_{vu} \qquad \forall l \in L,\ v \in V \tag{12}$$

$$a_i - a_j + |V| \cdot y^l_{ij} \leq |V| - 1 \qquad \forall l \in L,\ (i,j) \in A \tag{13}$$

$$\sum_{(u,v) \in A_0} t_{uv} \cdot y^l_{uv} \leq \hat{t} \qquad \forall l \in L \tag{14}$$

$$x_{vl} \in \{0,1\} \qquad \forall l \in L,\ v \in V \qquad (15)$$

$$y^l_{uv} \in \{0,1\} \qquad \forall l \in L,\ (u,v) \in A_0 \qquad (16)$$

$$1 \leq a_i \leq |V| \qquad \forall i \in V \qquad (17)$$

The objective function (4) maximizes the number of full truck loads picked up and delivered to the stations and thus, the total balance increase in the PBS, compare equation (3). Inequalities (5) state that every station visit is performed by at most one vehicle. By equalities (6) we explicitly define that every tour contains the same amount of pickup visits as delivery visits. Inequalities (7) 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 cluster the depot's starting node 0 has to have one outgoing arc (8), and similarly, the depot's target node $0'$ has to have one incoming arc (9). The arc selection variables are linked with the assignment variables as follows: Equalities (10) 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 (11) guarantee that each node $v \in V$ which is assigned to cluster $l \in L$ has to have one corresponding ingoing arc. Equalities (12) 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 (13) by computing an ordering of the nodes in variables a_i. Inequalities (14) guarantee that the routes for each vehicle lie within the allowed time budget $\hat{t}$. Finally, (15) to (17) 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 7. The approach, however, does not scale well to larger instances.

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 above model by equations (5)–(7), the Hamiltonian path problems are represented by equations (12)–(14), and the connections between the AP and Hamiltonian path problems are given by equation (10) and (11). 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.

5. Logic-based Benders decomposition

In 1962 Benders came up with his classical decomposition technique to solve large MIP problems [3]. 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. [12] 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. [27] 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* [24]. Typically, Benders cuts are here obtained via logical deduction. In several applications, in particular in the domain of scheduling, LBBD achieved remarkable results.

Hooker [23] presents a solution method applicable to generic scheduling problems where he models the master problem as a MIP and solves the subproblems by constraint programming (CP). Reported results on the LBBD outperform a pure MIP and and a pure CP approach. Harjunkoski and Grossmann [17] 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 [23].

A technique similar to the principles of LBBD are *combinatorial cuts*, cf. Codato and Fischetti [5].

In the following we show how LBBD is applied to our MIP for BBSS. Section 5.1 describes the master problem and states its MIP formulation, while Section 5.2 discusses the subproblem and proposes a corresponding solution approach. Section 5.3 shows how the master problem and subproblem interact and how the algorithm finally

yields an optimal solution.

5.1. Master problem

We decompose the model (4)–(17) from Section 4 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 [13] and also applied for BBSS by Schuijbroek et al. [39].

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 [23] 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 clusters' tour durations via the arborescence polytope, flow variables f^l_{uv} and arc selection variables $z^l_{uv} \in \{0,1\}$ for all vehicles $l \in L$ and arcs $(u,v) \in A_0$ are used.

Furthermore, we define β to be an upper bound on the maximal number of station visits per cluster respectively 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 7, 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 β.

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

$$\max \quad \sum_{l \in L} \sum_{v \in V} x_{vl} \tag{18}$$

$$\text{s.t.} \quad \sum_{v \in V} x_{vl} \leq \beta \qquad \forall l \in L \tag{19}$$

$$\sum_{l \in L} x_{vl} \leq 1 \qquad \forall v \in V \tag{20}$$

$$\sum_{v \in V_{\text{pic}}} x_{vl} = \sum_{v \in V_{\text{del}}} x_{vl} \qquad \forall l \in L \quad (21)$$

$$x_{s_i l} \geq x_{s_{i+1} l} \qquad \forall s \in S,\ l \in L,\ i = 1, \ldots, f_s - 1 \quad (22)$$

$$\sum_{(0,v) \in A_0} z^l_{0v} = 1 \qquad \forall l \in L \quad (23)$$

$$\sum_{(u,0') \in A_0} z^l_{u0'} = 1 \qquad \forall l \in L \quad (24)$$

$$z^l_{uv} \leq x_{ul} \qquad \forall l \in L,\ u \in V,\ (u,v) \in A_0 \quad (25)$$

$$z^l_{uv} \leq x_{vl} \qquad \forall l \in L,\ v \in V,\ (u,v) \in A_0 \quad (26)$$

$$\sum_{(0,v) \in A_0} f^l_{0v} = \sum_{v \in V} x_{vl} + 1 \qquad \forall l \in L \quad (27)$$

$$\sum_{(v,0') \in A_0} f^l_{v0'} = 1 \qquad \forall l \in L \quad (28)$$

$$\sum_{(u,v) \in A_0} f^l_{uv} - \sum_{(v,w) \in A_0} f^l_{vw} = x_{vl} \qquad \forall l \in L,\ v \in V \quad (29)$$

$$f^l_{uv} \leq \begin{cases} (\beta + 1) \cdot z^l_{0v} & \text{if } u = 0 \\ \beta \cdot z^l_{uv} & \text{if } v \in V_{\text{pic}} \\ (\beta - 1) \cdot z^l_{uv} & \text{else} \end{cases} \qquad \forall l \in L,\ (u,v) \in A_0 \quad (30)$$

$$\sum_{(u,v) \in A_0} z^l_{uv} = \sum_{v \in V} x_{vl} + 1 \qquad \forall l \in L \quad (31)$$

$$\sum_{(u,v) \in A_0} t_{uv} \cdot z^l_{uv} \leq \hat{t} \qquad \forall l \in L \quad (32)$$

$$x_{vl} \in \{0,1\} \qquad \forall l \in L,\ v \in V \quad (33)$$

$$z^l_{uv} \in \{0,1\} \qquad \forall l \in L,\ (u,v) \in A_0 \quad (34)$$

$$0 \leq f^l_{uv} \leq \begin{cases} \beta + 1 & \text{if } u = 0 \\ \beta & \text{if } v \in V_{\text{pic}} \\ \beta - 1 & \text{else} \end{cases} \qquad \forall l \in L,\ (u,v) \in A_0 \quad (35)$$

As in the compact model, the objective function (18) to be maximized is the total number of performed station visits. The maximum number of station visits per cluster are bounded upwards by β (19), the optimal solution or rounded down upper bound of the single-vehicle case, cf. appendix A. Inequalities (20) state that any station visit can only be performed by at most one vehicle. Equations (21) ensure that for every vehicle the number of assigned delivery station visits corresponds to the number of assigned pickup station visits. Inequalities (22) ensure that the $i+1$-th visit of a station can only be performed when an i-th visit takes place. Equalities (23) and (24) state that each vehicle leaves node 0 once and arrives at $0'$ once, respectively. Equalities (25) and (26) link the cluster assignment variables x_{vl} with the arc selection variables z^l_{uv}.

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.

The arborescence is realized by the single commodity flow conservation equations in equations (27)–(31). According to (27) 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 (28). Equalities (29) 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. Inequalities (30) link the flow variables with the arc selection variables z^l_{uv}, i.e., a positive flow may only occur on a selected arc. Equations (31) state that for each station visit assigned to vehicle l, exactly one incoming arc must be selected. Inequalities (32) 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 (33)–(35) are the domain definitions of the decision variables.

5.2. Subproblems

A solution to the master problem yields an assignment of stations to vehicles in variables x_{vl}. The *subproblem* (SP) in our LBBD corresponds then to the task of finding for each cluster/vehicle $l \in L$ in graph G_0 a Hamiltonian path from 0 to $0'$ visiting each node $v \in V \mid x_{vl} = 1$ 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* [7] 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 instance into an STSP instance by employing the method from Jonker et al. [25]. A symmetric auxiliary graph $G^{\text{aux}} = (V^{\text{aux}}, E^{\text{aux}})$ with associated costs $t^{\text{aux}} : V \to \mathcal{R}^+$ is derived. Its set of vertices consists of two nodes for each one in V and two nodes 0 and $0'$ representing the depot: $V^{\text{aux}} = \{v, v' \mid v \in V\} \cup \{0, 0'\}$. 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:

$$t^{\text{aux}}_{vv'} = t^{\text{aux}}_{v'v} = 0 \qquad \forall v \in V \cup \{0\} \tag{36}$$

$$t^{\text{aux}}_{u,v} = t^{\text{aux}}_{u',v'} = \infty \qquad \forall u, v \in V \cup \{0\},\ u \neq v \tag{37}$$

$$t^{\text{aux}}_{u,v'} = t^{\text{aux}}_{v'u} = t_{uv} \qquad \forall (u, v) \in A \tag{38}$$

$$t^{\text{aux}}_{u,v'} = t^{\text{aux}}_{v'u} = \infty \qquad \forall u, v \in V,\ u \neq v,\ (u, v) \notin A \tag{39}$$

$$t^{\text{aux}}_{0,v'} = t^{\text{aux}}_{v',0} = t_{0v} \qquad \forall v \in V_{\text{pic}} \tag{40}$$

$$t^{\text{aux}}_{0',v} = t^{\text{aux}}_{v,0'} = t_{v0'} \qquad \forall v \in V_{\text{del}} \tag{41}$$

Figure 1 shows the derivation of the auxiliary graph on an example. A TSP solution on graph G^{aux} will always connect a node $v \in V \cup \{0\}$ directly with its copy v' due to the

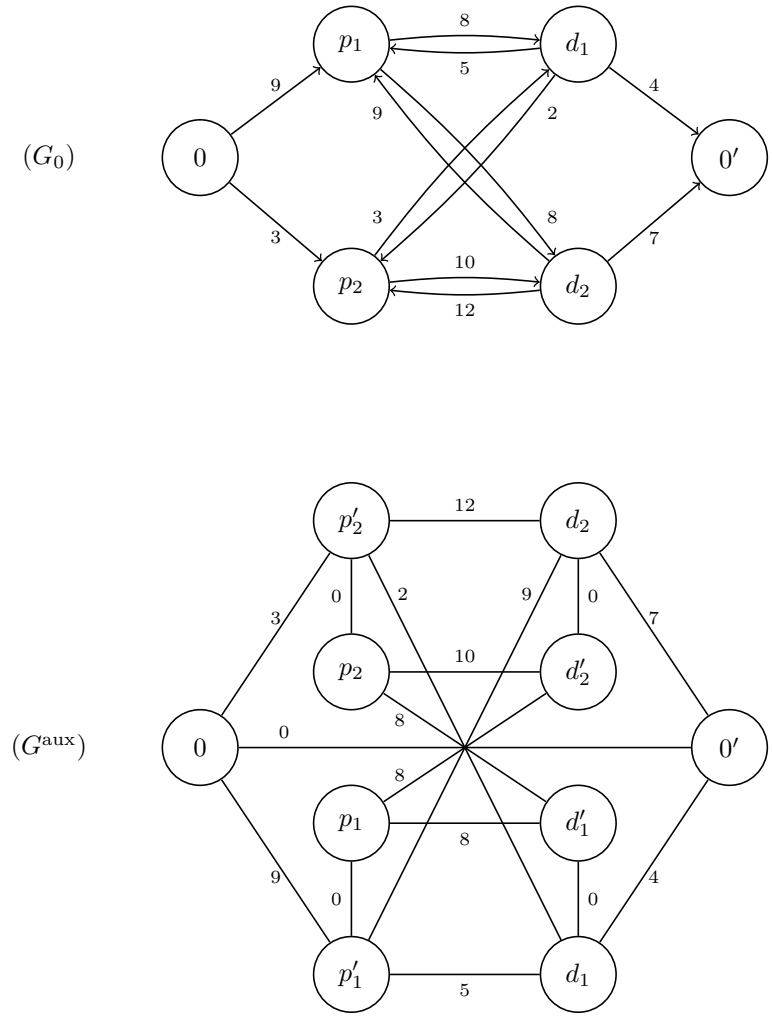


Figure 1: An example for the conversion of our subproblem on graph G_0 into a symmetric TSP instance on an auxiliary graph G^{aux}. Pickup stations are referred by p_1 and p_2, d_1 and d_2 denote delivery stations, 0 is the depot and $0'$ is the copy of the depot. Note, that G^{aux} actually is a complete graph. However, infeasible edges with $t_{uv} = \infty$, $\forall (u, v) \in G^{\text{aux}}$ are omitted for the sake of readability.

zero-cost edges, 0 to a node $v \in V_{\text{pic}}$, a node u' with $u \in V_{\text{pic}}$ with a node $v \in V_{\text{del}}$, and a node u' with $u \in V_{\text{del}}$ with a node v' with $v \in V_{\text{pic}} \cup \{0\}$. Consequently, a minimum duration Hamiltonian path from 0 to $0'$ can be obtained from an optimal TSP tour on G^{aux} by taking for each selected edge (u, v') the arc (u, v), for all $u \in V \cup \{0\}$ and $v \in V$ and arc $(u, 0')$ for edge $(v, 0')$.

5.3. Iterated Decomposition Procedure and Cut Generation

Algorithm 1 LBBD for BBSS

```
1: repeat
2:    init: r ← vector of |L| empty routes,  cutsAdded ← false
3:    Solve MP to obtain subproblems
4:    for all l ∈ L do
5:       r_l ← solve SP for vehicle l
6:       if r_l^obj > t̂ then
7:          I ←Try to minimize infeasible set of station visits in r_l
8:          MP ← MP ∪ (Σ_{v∈I} x_vl ≤ |I| − 1   ∀l ∈ L)
9:          cutsAdded ← true
10:      end if
11:   end for
12: until not(cutsAdded)
13: return r
```

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 shorthand notation r_l^{obj} stands for the objective value of a single subproblem solution, i.e., the actual routing costs when the TSP is solved to optimality.

The master problem 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). In this case an infeasibility cut of the form

$$\sum_{v \in I} x_{vl} \leq |I| - 1 \tag{42}$$

is created for each vehicle $l \in L$ and added to the master problem, where I is a set of station visits. These cuts imply that the simultaneous assignment of the station

visits in I – and all supersets of I – to one cluster is prohibited in the following 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 the following algorithm:

Algorithm 2 Minimize the cutset

init: $MinSize \leftarrow |I|,\ T = \emptyset$
function $minimizeCutSet(r_l)$
1: **for all** $(u, v) \in r_l \mid u \notin \{0, 0'\}, v \notin \{0, 0'\}$ **do**
2: $T \leftarrow$ extract all nodes from r_l except u and v
3: $r'_l \leftarrow$ solve SP for stations in T
4: **if** $r_l'^{\text{obj}} > \hat{t}$ **then**
5: **if** $|T| = MinSize$ **then**
6: $I \leftarrow I \cup \{T\}$
7: **else if** $|T| < MinSize$ **then**
8: $I \leftarrow \emptyset$
9: $I \leftarrow I \cup \{T\}$
10: $MinSize \leftarrow |T|$
11: **end if**
12: $minimizeCutSet(r'_l)$
13: **end if**
14: **end for**

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 all nodes from the Hamiltonian path except u and v and refer this set as T (2). Two stations 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 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* (12) 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.

5.3.1. Cluster-spanning cuts

Due to the following observation we came up with the idea of also computing cluster-spanning cuts instead of only utilizing cuts for single clusters: Consider, if some of the computed clusters from the master problem are infeasible, we generate a cut for every single cluster. In a new iteration of the master problem the MIP solver often tries to move station visits among different clusters – because then these new assignments constitute different clusters 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 clusters 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 cluster-spanning cuts can be generated. Thus, we take the reduced set of station visits as solution from the master problem but breakup the assignment to the clusters and compute the minimal routing costs resulting from an optimal assignment. We therefore adjusted inequalities (20) 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 clusters, 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 stations 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 cluster-spanning cut iff

$$\sum_{l \in L} h_l > |L| \cdot \hat{t}. \tag{43}$$

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 (43) holds for this assignment. Then formally, the cut is defined as follows:

$$\sum_{l \in L} \sum_{v \in I} x_{vl} \leq |I| - 1 \tag{44}$$

5.4. Branch-and-Check

As an alternative to the classical (logic-based) Benders decomposition approach followed in the previous section, we also consider a corresponding *BAC* approach. The term BAC has been originally proposed by Thorsteinsson in [41] and refers to the following. Instead of completely resolving the master problem after adding new Benders cuts in each iteration, the master problem is now only solved once, and Benders cuts are separated in a branch-and-cut manner whenever a solution supposed to be feasible is encountered.

Thus, the Benders subproblem is solved not only for so-far optimal master problem solutions, but for any encountered feasible master problem solution. Again, Concorde is used for solving the subproblems and algorithm 2 for finding minimal infeasible sets of station visits is applied to obtain possibly strengthened Benders cuts.

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 feasibility cuts is only able to provide lower bounds earlier than at the very end, as the master problem variables will not get overall feasible assignments before.

6. 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. [35]. 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 ω^{work} and ω^{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.

7. Computational Results

We have done our computational tests on a rigorous benchmark suite derived from [35] with instances up to 70 and 120 stations for the multi-vehicle and the single-vehicle cases. 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++ 4.9.2. As MIP solver we used CPLEX 12.6.2 branch-and-cut with default parameters except limiting the number of used threads to one for a better comparability and restricting the maximum size of the branch-and-cut tree to 3GB. 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 2 hours. 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.

The instances state specific initial fill levels p_s and target fill levels q_s for each station $s \in S$ as well as a uniform vehicle capacity Z. 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

$$f_s = \left\lfloor \frac{p_s - q_s}{Z} \right\rfloor \forall s \in S_{\text{del}} \quad \text{and} \quad f_s = \left\lceil \frac{p_s - q_s}{Z} \right\rceil \forall s \in S_{\text{pic}}. \tag{45}$$

Remember that delivery stations S_{del} have positive demand values f_s, while pickup stations are given by negative 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.

7.1. 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.

The MIP models available to us for details of the original problem formulation are, unfortunately, only able to exactly solve instances with up to 20 nodes, see [35]. Therefore, we consider here one of the leading metaheuristic approaches, which is the VNS from [34] introduced in Section 6.

Table 1 shows comparative results for instances which our BAC could solve to proven optimality. 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 by dividing it by the vehicle capacity Z to make it comparable to the number of full vehicle loads by which we measure the balance gain in our new model.

We show the number of instances where the particular approach yielded the best solution #best, the number of runs where BAC terminated with the optimal solution #opt, mean of the objective value $\overline{obj}$ over 30 instances per benchmark set, the standard deviation of the objective value sd_obj and the median of the runtimes $\widetilde{time}$. If BAC was not able to find the optimal solution we use the lower bound, i.e., objective value of the best identified solution. Only in very few cases, on larger instances, we have not been able to retrieve any feasible solution by BAC (12 out of 1260 instances) which explains the relatively large standard deviations on the instance groups $(70, 2, 480)$ and $(70, 3, 480)$. Note that BAC is only able to yield the best solution if the solution obtained via VNS is not optimal.

When analyzing the results in Table 1 one can see that the average objective values obtained by the BAC and the VNS correspond closely. Obviously, the simplification of considering only full vehicle loads has on these instances only a very minor impact.

Table 1: Comparison of BAC for our new model restricted to full vehicle loads with the VNS allowing fine-grained bike movements.

Instance set			**BAC**					**VNS**			
$\|V\|$	$\|L\|$	$\hat{t}$	#best	#opt	$\overline{obj}$	sd_obj	$\widetilde{time}$	#best	$\overline{obj}$	sd_obj	$\widetilde{time}$
10	2	120	**26**	30	6.80	1.5403	0.10	**26**	6.87	1.2521	3600
10	2	240	**30**	30	8.87	1.2521	0.10	**30**	8.87	1.2521	3600
10	2	480	**30**	30	8.87	1.2521	0.10	**30**	8.87	1.2521	3600
10	3	120	26	30	8.13	1.5698	0.10	**30**	8.40	1.3287	3600
10	3	240	**30**	30	8.87	1.2521	0.10	**30**	8.87	1.2521	3600
10	3	480	**30**	30	8.87	1.2521	0.10	**30**	8.87	1.2521	3600
20	2	120	**30**	30	7.80	0.6103	0.40	28	7.67	0.9223	3600
20	2	240	**28**	30	16.40	1.6938	2.00	26	16.40	1.3287	3600
20	2	480	**30**	30	18.87	2.3887	0.40	**30**	18.87	2.3887	3600
20	3	120	**30**	30	11.20	1.3493	1.40	26	10.93	1.6386	3600
20	3	240	**30**	30	18.87	2.3887	1.00	**30**	18.87	2.3887	3600
20	3	480	**30**	30	18.87	2.3887	0.70	**30**	18.87	2.3887	3600
30	2	120	28	30	7.67	1.0613	0.90	**30**	7.93	0.3651	3600
30	2	240	**30**	24	18.53	1.0417	241.50	20	17.87	1.0417	3600
30	2	480	**30**	30	28.87	2.9094	6.20	**30**	28.87	2.9094	3600
30	3	120	**28**	29	11.40	1.6733	2.20	**28**	11.67	0.9223	3600
30	3	240	**30**	12	25.47	1.7367	3600.00	23	25.00	1.4622	3600
30	3	480	**30**	30	28.87	2.9094	8.40	**30**	28.87	2.9094	3600
40	2	120	29	30	7.87	0.7303	2.50	**30**	8.00	0.0000	3600
40	2	240	**27**	22	18.80	1.4479	296.90	21	18.53	1.2794	3600
40	2	480	**30**	16	35.93	1.5298	530.30	23	35.47	1.2794	3600
40	3	120	**29**	30	11.67	1.1842	5.40	28	11.73	0.6915	3600
40	3	240	**27**	9	26.47	1.6344	3600.00	23	26.33	1.1842	3600
40	3	480	**30**	30	38.33	2.7334	15.10	**30**	38.33	2.7334	3600
50	2	120	**30**	30	7.93	0.3651	4.60	**30**	7.93	0.3651	3600
50	2	240	**30**	26	19.53	1.1366	115.00	26	19.27	1.3374	3600
50	2	480	**29**	1	38.67	1.6046	3600.00	17	37.87	1.7367	3600
50	3	120	**30**	29	11.80	0.8052	18.70	**30**	11.80	0.8052	3600
50	3	240	**28**	9	27.67	1.9711	3600.00	20	27.07	1.5522	3600
50	3	480	**30**	27	48.73	2.7029	101.90	27	48.53	2.4598	3600
60	2	120	**30**	30	8.00	0.0000	10.50	**30**	8.00	0.0000	3600
60	2	240	**30**	23	19.60	0.8137	570.50	21	19.00	1.1447	3600
60	2	480	**30**	0	39.33	1.6046	3600.00	15	38.33	1.3979	3600
60	3	120	**30**	29	11.93	0.3651	31.80	28	11.80	0.6103	3600
60	3	240	**24**	6	27.07	2.0833	3600.00	22	26.93	1.3629	3600
60	3	480	**29**	3	53.67	1.3979	3600.00	22	53.20	1.4479	3600
70	2	120	**30**	30	8.00	0.0000	22.20	**30**	8.00	0.0000	3600
70	2	240	**27**	23	19.60	0.9685	522.40	**27**	19.60	0.8137	3600
70	2	480	**27**	0	39.07	7.4599	3600.00	16	39.53	1.5477	3600
70	3	120	**30**	29	11.93	0.3651	68.40	**30**	11.93	0.3651	3600
70	3	240	22	9	27.67	2.1709	3600.00	**24**	27.93	1.5298	3600
70	3	480	17	0	35.33	27.3790	3600.00	**20**	55.13	1.5477	3600

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. 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 11 of the 42 classes whereas the VNS has significant advantages on 2 of the 42 classes.

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

7.2. Gains of Logic-Based Benders Decomposition Compared to Compact Model

In Table 2 a comparison between LBBD approach and applying CPLEX directly to the compact model is shown.

Column *opt* shows the percentage of instances solved to optimality out of the 30 different instances of each instance group. $\overline{obj}$ shows the average number of stations that can be serviced within the given time budget $\hat{t}$ aggregated over the instance group. By *mem* we show how many of the instances did not succeed because of exceeding the memory limit of 3GB. The median of the time is shown in the column $\widetilde{time}$ and is given in seconds. We marked those values bold where the corresponding approach managed to solve more instances to optimality with respect to the others.

Most of the small instances with 10 and 20 stations are solved to proven optimality by both approaches in a very small amount of time. On larger instances the LBBD approach relatively clearly outperforms the compact approach with respect to the number of instances that could be solved to optimality. We notice that CPLEX needs a lot of memory for the compact MIP model because of its high size in constraints and variables. This is much better in the decomposition approach because the master and the subproblem are both small models and subproblems do not have to be maintained, so they can be discarded once they are solved. For the master problem we rely on the resolve feature of CPLEX where it tries with repair heuristics to recover from the last solution found.

Of course, depending on the structure of the instance either the LBBD or the compact model is more likely to be successful. But overall, the LBBD could solve 151 more instances to optimality than the compact model. This is also due to the fact that on larger instances the performance of the LBBD gets better with respect to the compact model.

7.3. Enhancements by Branch-and-Check

In Table 3 we show a comparison between the BAC approach on the left side and the LBBD on the right side. In contrast to the previous table we also give the average number of iterations for the LBBD denoted by column header $\overline{iter}$ and the number of calls to LazyConstraintCallback of the BAC approach denoted by $\overline{calls}$. By $\overline{cuts}$, we

Table 2: Comparison of LBBD with the compact model

Instance set			LBBD			Compact model			
$\|V\|$	$\|L\|$	$\hat{t}$	*opt* [%]	$\overline{obj}$	$\widetilde{time}[s]$	*opt* [%]	$\overline{obj}$	*mem* [%]	$\widetilde{time}[s]$
10	2	120	**100**	6.80	0.14	**100**	6.73	0	0.04
10	2	240	**100**	8.87	0.05	**100**	8.07	0	0.02
10	2	480	**100**	8.87	0.05	**100**	8.07	0	0.01
20	2	120	**100**	7.80	0.35	**100**	7.80	0	0.12
20	2	240	**100**	16.40	1.66	**100**	16.40	0	0.43
20	2	480	**100**	18.87	0.14	**100**	17.73	0	0.09
20	3	120	**100**	11.20	1.36	97	10.80	0	0.70
20	3	240	**100**	18.87	0.45	**100**	17.73	0	0.14
20	3	480	**100**	18.87	0.17	**100**	17.73	0	0.10
30	2	120	**100**	7.67	0.84	97	7.67	0	0.54
30	2	240	**67**	12.40	941.38	**67**	12.33	13	84.02
30	2	480	**100**	28.87	0.99	**100**	27.73	0	0.41
30	3	120	**97**	11.13	1.84	77	9.00	3	106.41
30	3	240	27	6.53	3600.00	**67**	16.80	13	126.26
30	3	480	**100**	28.87	1.07	**100**	27.73	0	0.49
40	2	120	**100**	7.87	3.06	80	6.40	3	2.93
40	2	240	**73**	14.07	250.32	53	10.53	13	583.66
40	2	480	53	19.20	1506.95	**93**	33.13	0	5.01
40	3	120	**97**	11.33	4.81	43	5.20	13	1942.29
40	3	240	20	5.20	3600.00	**33**	9.40	43	709.20
40	3	480	**100**	38.33	2.92	**100**	37.20	0	1.26
50	2	120	**100**	7.93	3.84	63	4.93	7	184.78
50	2	240	**87**	17.07	149.02	47	9.33	20	498.81
50	2	480	3	1.20	3600.00	**47**	18.47	37	604.39
50	3	120	**93**	11.20	17.12	13	1.60	30	3591.50
50	3	240	**27**	7.60	3600.00	0	0.00	63	1078.43
50	3	480	90	43.67	27.93	**93**	44.33	3	3.13
60	2	120	**100**	8.00	10.63	67	5.33	7	20.62
60	2	240	**73**	14.67	326.49	43	8.67	23	735.88
60	2	480	0	0.00	3600.01	**20**	8.00	60	672.99
60	3	120	**97**	11.60	31.29	27	3.20	17	3588.24
60	3	240	**17**	5.00	3600.01	3	1.00	63	835.63
60	3	480	13	7.33	3600.01	**20**	10.67	57	1122.85
70	2	120	**100**	8.00	14.31	47	3.73	13	836.50
70	2	240	**77**	15.33	324.55	47	9.47	27	852.23
70	2	480	**3**	1.33	3600.01	0	0.00	63	988.07
70	3	120	**97**	11.60	48.43	23	2.80	27	3268.72
70	3	240	**30**	9.00	3600.01	0	0.00	57	1352.00
70	3	480	**0**	0.00	3600.01	**0**	0.00	73	1133.34

also provide the number of generated Benders infeasibility cuts for both approaches. We also give the median of the maximum time needed to solve the master problems $\widetilde{max\_master}$ and the average time needed to solve the master problem respectively $\widetilde{avg\_master}$. BAC was only in two instance sets worse than LBBD but in all other cases it was equal or even better. With respect to the number of optimal solutions retrieved, BAC outperformed LBBD on 8 instance sets. As it can be seen, the number of generated cuts are higher when using BAC because the feasibility check in the subproblem is done for each feasible solution whereas for the LBBD cuts are only generated when the master problem is solved to optimality. However, as shown more generated Benders cuts do not necessarily yield better performance. Of course, it depends also on the quality of the cuts.

Table 3: Comparison of BAC with LBBD

Instance set			**BAC**						**LBBD**						
$\|V\|$	$\|L\|$	$\hat{t}$	*opt* [%]	*obj*	$\overline{calls}$	$\overline{cuts}$	*mem* [%]	$\widetilde{time}$ [*s*]	*opt* [%]	$\overline{obj}$	$\overline{iter}$	$\overline{cuts}$	$\widetilde{max\_master}$ [*s*]	$\widetilde{avg\_master}$ [*s*]	$\widetilde{time}$ [*s*]
20	2	480	**100**	18.87	1.13	0.00	0	0.44	**100**	18.87	1.00	0.00	0.00	0.00	0.14
20	3	480	**100**	18.87	1.07	0.00	0	0.69	**100**	18.87	1.00	0.00	0.00	0.00	0.17
30	2	120	**100**	7.67	2.37	1.80	0	0.94	**100**	7.67	1.70	1.53	0.00	0.00	0.84
30	2	240	**80**	14.87	54.80	158.90	0	241.45	67	12.40	41.63	142.00	105.00	26.50	941.38
30	2	480	**100**	28.87	2.00	1.07	0	6.24	**100**	28.87	1.03	0.07	0.00	0.00	0.99
30	3	120	**97**	11.13	3.90	7.13	0	2.21	**97**	11.13	3.20	10.57	1.00	1.00	1.84
30	3	240	**40**	10.27	34.97	171.53	0	3600.00	27	6.53	6.77	40.73	1707.50	715.50	3600.00
30	3	480	**100**	28.87	1.17	0.00	0	8.45	**100**	28.87	1.00	0.00	0.00	0.00	1.07
40	2	120	**100**	7.87	2.53	2.13	0	2.52	**100**	7.87	2.20	3.00	1.00	1.00	3.06
40	2	240	**73**	14.00	67.27	256.77	0	296.88	**73**	14.07	21.83	87.57	27.00	10.50	250.32
40	2	480	**53**	19.27	66.17	268.73	7	530.35	**53**	19.20	8.63	27.67	268.00	113.50	1506.95
40	3	120	**100**	11.67	4.77	12.27	0	5.42	97	11.33	3.43	13.10	4.00	3.00	4.81
40	3	240	**30**	8.07	46.57	228.50	0	3600.00	20	5.20	4.13	26.73	2276.50	1437.50	3600.00
40	3	480	**100**	38.33	1.63	0.10	0	15.12	**100**	38.33	1.03	0.10	1.00	1.00	2.92
50	2	120	**100**	7.93	2.57	2.80	0	4.61	**100**	7.93	2.47	4.20	3.00	2.00	3.84
50	2	240	**87**	17.07	32.80	81.20	0	115.00	**87**	17.07	8.10	25.97	32.00	10.00	149.02
50	2	480	**3**	1.20	59.70	223.53	13	3600.01	**3**	1.20	12.97	56.63	1638.50	577.50	3600.00
50	3	120	**97**	11.53	7.57	20.27	0	18.74	93	11.20	2.93	9.43	9.50	8.00	17.12
50	3	240	**30**	8.60	45.40	212.33	3	3600.01	27	7.60	7.07	44.23	1088.50	412.50	3600.00
50	3	480	**90**	43.67	6.33	16.00	0	101.93	**90**	43.67	1.40	2.10	7.00	7.00	27.93
60	2	120	**100**	8.00	3.10	3.23	0	10.46	**100**	8.00	2.37	3.70	6.50	5.00	10.63
60	2	240	**77**	15.27	53.37	136.50	0	570.52	73	14.67	9.33	33.40	83.50	39.00	326.49
60	2	480	**0**	0.00	55.27	155.27	10	3600.01	**0**	0.00	6.33	28.77	1864.00	735.00	3600.01
60	3	120	**97**	11.60	8.73	25.00	0	31.78	**97**	11.60	2.83	8.17	17.50	14.50	31.29
60	3	240	**20**	6.00	35.13	145.10	7	3600.01	17	5.00	3.27	19.63	2491.50	1796.00	3600.01
60	3	480	10	5.40	27.27	124.83	0	3600.01	**13**	7.33	0.60	2.80	3579.50	3579.50	3600.01
70	2	120	**100**	8.00	5.70	10.07	0	22.15	**100**	8.00	1.60	1.50	12.00	11.00	14.31
70	2	240	**77**	15.33	40.83	91.23	3	522.43	**77**	15.33	6.63	19.67	96.50	44.00	324.55
70	2	480	0	0.00	29.83	82.34	0	3600.01	**3**	1.33	4.50	19.77	2049.00	1140.50	3600.01
70	3	120	**97**	11.60	8.17	23.47	0	68.36	**97**	11.60	2.63	7.90	30.50	29.50	48.43
70	3	240	**30**	9.00	30.50	121.43	7	3600.01	**30**	9.00	2.73	13.50	2510.00	1491.00	3600.01
70	3	480	**0**	0.00	13.89	50.68	0	3600.01	**0**	0.00	0.17	1.63	3360.50	3360.50	3600.01

7.4. Single-Vehicle Case

Table 4: Computational results for the single-vehicle case

$\|V\|$	$\hat{t}$	opt [%]	$\overline{obj}$	$\overline{UB}$	$\overline{LB}$	$\overline{gap}$	mem [%]	$\widetilde{time}$
40	240	97	9.60	9.93	9.98	0.45	0	0.39
40	480	100	20.40	20.40	20.40	0.00	0	0.94
50	120	100	4.00	4.00	4.00	0.00	0	0.21
50	240	93	9.33	9.93	10.03	1.08	0	1.01
50	480	90	19.27	21.27	21.42	0.74	0	4.83
60	120	100	4.00	4.00	4.00	0.00	0	0.26
60	240	83	8.40	10.00	10.45	4.65	0	1.52
60	480	87	18.73	21.40	21.73	1.58	0	86.13
70	120	100	4.00	4.00	4.00	0.00	0	0.39
70	240	83	8.47	10.13	10.46	3.26	0	2.93
70	480	67	14.67	21.67	22.18	2.46	0	27.36
90	120	100	4.00	4.00	4.00	0.00	0	0.62
90	240	80	10.08	10.07	10.35	2.86	0	16.36
90	480	63	22.21	22.14	23.41	5.78	0	38.41
120	120	100	3.93	3.93	3.93	0.00	0	1.35
120	240	57	5.67	10.00	10.64	6.43	0	2302.56
120	480	37	8.20	22.13	23.23	5.01	13	3600.00

We have also performed computational tests on the single-vehicle case of the problem. 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 8 hours for the driver we have been able to solve 37% of the instances to proven optimality. For the remaining instances we have been able to provide results with an average optimality gap of about 5%.

8. 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 11 of the 42 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.

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] 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.

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

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

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

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

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

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

[8] 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.

[9] P. DeMaio, Bike-sharing: History, impacts, models of provision, and future, Public Transportation 12 (2009), 41–56.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

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

[26] 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.

[27] 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).

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] B. Schroeder, Bicycle sharing 101: Getting the wheels turning, Moonshine Media, January 2014.

[39] J. Schuijbroek, R. Hampshire, and W.J. van Hoeve, Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems, Technical report 2013-E1, Tepper School of Business, Carnegie Mellon University, 2013.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

A. Single-vehicle MIP model

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

$$\max \sum_{v \in V} x_v \tag{46}$$

$$\text{s.t.} \quad x_v \leq 1 \qquad \forall v \in V \tag{47}$$

$$\sum_{v \in V_{\text{pic}}} x_v = \sum_{v \in V_{\text{del}}} x_v \tag{48}$$

$$x_{s_i} \geq x_{s_{i+1}} \qquad \forall s \in S, i = 1, \ldots, f_s - 1 \tag{49}$$

$$\sum_{v \in V_{\text{pic}}} y_{0v} = 1 \tag{50}$$

$$\sum_{v \in V_{\text{del}}} y_{v0'} = 1 \tag{51}$$

$$\sum_{(u,v) \in A_0} y_{uv} = x_u \qquad \forall u \in V \tag{52}$$

$$\sum_{(u,v) \in A_0} y_{uv} = x_v \qquad \forall v \in V \tag{53}$$

$$\sum_{(u,v) \in A_0} y_{uv} = \sum_{(v,u) \in A_0} y_{vu} \qquad \forall v \in V \tag{54}$$

$$a_i - a_j + |V| \cdot y_{ij} \leq |V| - 1 \qquad \forall (i,j) \in A \tag{55}$$

$$\sum_{(u,v) \in A_0} y_{uv} \cdot t_{uv} \leq \hat{t} \tag{56}$$

$$x_v \in \{0,1\} \qquad \forall v \in V \tag{57}$$

$$y_{uv} \in \{0,1\} \qquad \forall (u,v) \in A_0 \tag{58}$$

$$1 \leq a_i \leq |V| \qquad \forall i \in V \qquad (59)$$

B. Routing MIP model

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

$$\min \sum_{l \in L} \sum_{(u,v) \in A_0} y^l_{uv} \cdot t_{uv} \qquad (60)$$

$$\text{s.t.} \quad \sum_{l \in L} x_{vl} = 1 \qquad \forall v \in V \qquad (61)$$

$$\sum_{v \in V_{\text{pic}}} x_{vl} = \sum_{v \in V_{\text{del}}} x_{vl} \qquad \forall l \in L \qquad (62)$$

$$x_{s_i l} \geq x_{s_{i+1} l} \qquad \forall s \in S, l \in L, i = 1, \ldots, f_s - 1 \qquad (63)$$

$$\sum_{v \in V_{\text{pic}}} y^l_{0v} = 1 \qquad \forall l \in L \qquad (64)$$

$$\sum_{v \in V_{\text{del}}} y^l_{v0'} = 1 \qquad \forall l \in L \qquad (65)$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{ul} \qquad \forall l \in L, u \in V \qquad (66)$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = x_{vl} \qquad \forall l \in L, v \in V \qquad (67)$$

$$\sum_{(u,v) \in A_0} y^l_{uv} = \sum_{(v,u) \in A_0} y^l_{vu} \qquad \forall l \in L, v \in V \qquad (68)$$

$$a_i - a_j + |V| \cdot y^l_{ij} \leq |V| - 1 \qquad \forall l \in L, (i,j) \in A \qquad (69)$$

$$x_{vl} \in \{0, 1\} \qquad \forall l \in L, v \in V \qquad (70)$$

$$y^l_{uv} \in \{0, 1\} \qquad \forall l \in L, (u,v) \in A_0 \qquad (71)$$

$$1 \leq a_i \leq |V| \qquad \forall i \in V \qquad (72)$$