Submitted to *Transportation Science*
manuscript

The static bike sharing rebalancing problem with forbidden temporary operations

Bruno P. Bruck
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Italy, bruno.petratobruck@unimore.it
Centro de Informática, Universidade Federal da Paraíba, Brazil, bruno.bruck@ci.ufpb.br

Fábio Cruz
Centro de Informática, Universidade Federal da Paraíba, Brazil, fabiocbalbuquerque@gmail.com

Manuel Iori
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Italy, manuel.iori@unimore.it

Anand Subramanian
Centro de Informática, Universidade Federal da Paraíba, Brazil, anand@ci.ufpb.br

This paper introduces and solves the static bike rebalancing problem with forbidden temporary operations. In this problem, one aims at finding a minimum cost route in which a vehicle performs a series of pickup and delivery operations, while satisfying demand and capacity constraints. In addition, a vehicle can visit stations multiple times but cannot use them to temporary store or provide bikes. Apart from bike rebalancing, the problem also models courier service transportation and repositioning of inventory between retail stores, where temporary operations are frequently disliked because they require additional manual work and service time. We present some theoretical results concerning problem complexity and worst case analysis, and then propose three exact algorithms based on different mathematical formulations. Extensive computational results on instances involving up to 80 stations show that an exact algorithm based on a minimal extended network produces the best average results.

Key words: Bike sharing; branch-and-cut; minimal extended network

1. Introduction

In the *static bike rebalancing problem with forbidden temporary operations* (SBRP-FT), we are given a directed graph $G = (V, A)$, where $V = I \cup \{0\}$ is the set of vertices, $I = \{1, ..., n\}$ is the set of stations, 0 is the depot, and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. Each arc $(i, j) \in A$ is associated with a traveling cost c_{ij}, possibly asymmetric but satisfying the triangle inequality. Each station $i \in V$ has a demand d_i of bikes, and is said to be in *excess* if $d_i > 0$, in *default* if $d_i < 0$, or

initially balanced if $d_i = 0$. We assume that the system is balanced (i.e., $\sum_{i \in I} d_i = 0$) and the depot has no demand (i.e., $d_0 = 0$).

The SBRP-FT aims at finding a minimum cost route in which a vehicle performs a series of pickup and delivery operations to satisfy the demand of each station, while ensuring that the following constraints are not violated: (i) the route starts and ends at the depot with the vehicle empty; (ii) the vehicle capacity is not exceeded (and negative loads are not allowed) at any point of the route; (iii) the demand of each station is completely fulfilled by using one or more vehicle visits; (iv) and no temporary operations are performed. The last constraint imposes that the vehicle cannot use stations as depots to either temporarily dropoff bikes (and collect them later on the route) or temporarily pickup bikes (and drop them off later on).

Furthermore, split of demands at unbalanced stations is allowed and, because we assume the triangle inequality to hold over c and temporary operations to be forbidden, we also assume that initially balanced stations need not be visited. Notice that, the latter assumption is not strictly necessary, and might be removed in scenarios where all stations must be visited (i.e., for daily inspections to the stations). In addition, note that initial unbalanced systems might be taken into account by simply adding a dummy station positioned at the same coordinates of the depot and having demand equal to $-\sum_{i \in I} d_i$ (as suggested in Hernández-Pérez and Salazar-González 2004a).

The SBRP-FT is of combinatorial interest not only because it is an $\mathcal{NP}$-hard problem, as the *traveling salesman problem* (TSP) is a special case of it, but it is also very challenging in practice. Allowing split demands and, at the same time, forbidding temporary operations, significantly increases the complexity of the problem and of developing efficient and robust exact algorithms. In addition, the SBRP-FT is a general problem whose interest is not limited to bike sharing systems. It can be used, indeed, to model many other contexts, such as courier service transportation and the repositioning of inventory between retail stores, in the case where temporary operations are forbidden. Temporary dropoff/pickup operations require additional manual work and service time for loading and unloading, as well as additional risk of damaging the products, and are thus frequently forbidden by transportation operators.

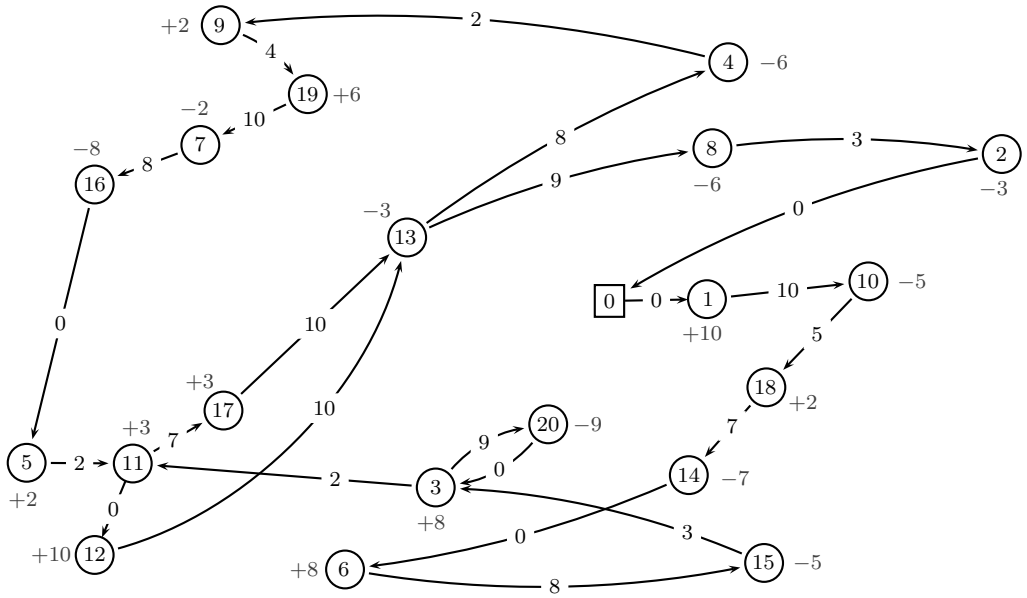
In the context of bike repositioning, forbidding temporary operations is a typical constraint imposed by system managers that aims at alleviating drivers' working activities by not imposing too many loading/unloading operations (see Dell'Amico et al. 2014). In other cases, this constraint might originate from stronger considerations, and temporary dropoff operations might be forbidden in locations that do not fulfill certain technical requirements. This happens, for example, in the distribution of medicinal products, in which particular attention should be paid to temperature monitoring, cleanliness and the security of any intermediate storage facilities (Official Journal of the European Union 2013), thus imposing the installation of costly equipment. Another example can be found in the

transport of chemical products, where temporary dropoffs might require the presence of additional protective clothing for drivers and are allowed only at sites that meet the requirements established by regulations on risk mitigation (see Cefic–The European Chemical Industry Council 2013).

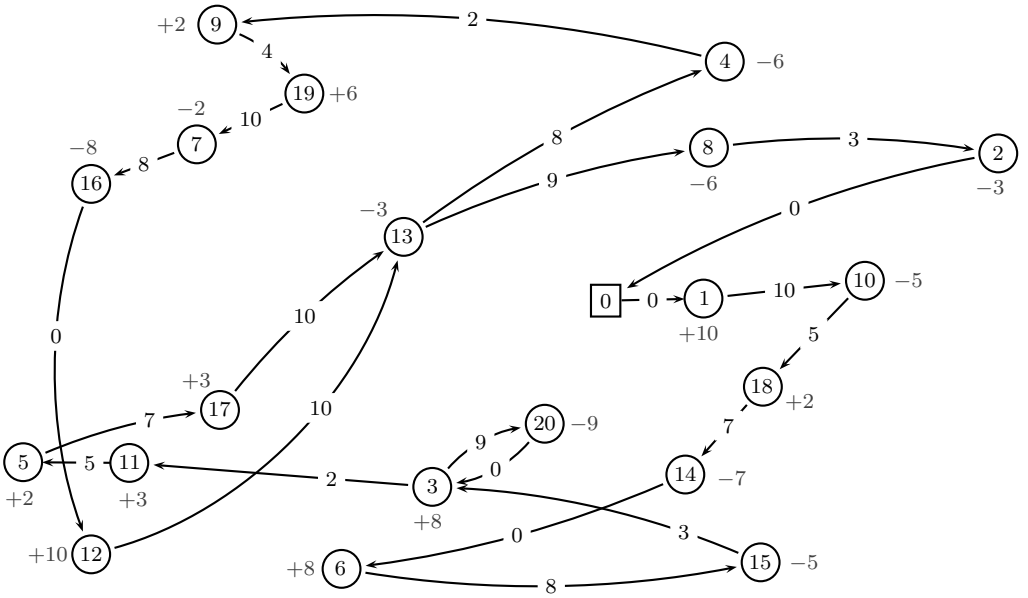
To our knowledge, this is the first work to directly approach the SBRP-FT. However, its closest variant, called the *static bicycle rebalancing problem* (SBRP), where temporary operations are allowed, has already been studied by Chemla, Meunier, and Wolfer Calvo (2013), Erdoğan, Battarra, and Wolfler Calvo (2015), and Cruz et al. (2017). Salazar-González and Santos-Hernández (2015) studied a general one-commodity pickup and delivery problem with split demands, and described how their model could be modified to forbid temporary operations and impose zero load on the vehicle when leaving the depot. Another realistic application close to the SBRP-FT is the *bike sharing rebalancing problem*, studied by Dell'Amico et al. (2014), where multiple vehicles are used but multiple visits and demand splits are not allowed. Details about these works are provided in the following section. Note that all the methods that we propose could easily deal with the cases in which demand splits are not allowed (by simply imposing each vertex to be visited once), or a service time is imposed on each delivery (by adding the service time to the traversal time of the arcs).

By forbidding temporary pickups and deliveries, the complexity of operations at stations is reduced, but routing costs might be higher than in the SBRP. As an illustrative example, consider Figure 1-(a), which shows an optimal solution for the SBRP on a benchmark instance. Each arc is traversed exactly once and the flow of commodity passing through the arc is shown over the arc. Each station is associated univocally with a vertex and the station demand is shown nearby. In the first visit to station 11, coming from station 3, instead of performing a pickup operation the vehicle temporarily deliveries 2 bikes and then continues with an empty load to station 12. In the second visit, coming from station 5, the vehicle fulfills the demand of station 11 and recollects the 2 units delivered during the first visit, thus proceeding to station 17 with a load of 7 bikes. Figure 1-(b) presents an optimal solution for the same instance, but for the SBRP-FT. Note that forbidding temporary operations results in a small increase in routing costs, from 6012 to 6025, but it also substantially reduces the complexity of operations at station 11.

In this paper, we propose three exact algorithms for the SBRP-FT, all making use of a *branch-and-cut* (B&C) framework, but with different emphases. The first two are based on the use of an aggregate formulation derived from the work of Chemla, Meunier, and Wolfer Calvo (2013), where an integer variable expresses the number of times the vehicle passes through an arc. The solution of this formulation may be infeasible for the SBRP-FT because temporary operations might be performed in stations visited twice or more. Thus, our first algorithm iteratively removes infeasible solutions, one at a time, in a process that we call *branch-and-reject*. Our second algorithm is a direct extension



(a) Optimal solution with temporary operations with value 6012 for instance n20C with $Q = 10$



(b) Optimal solution with forbidden temporary operations with value 6025 for instance n20C with $Q = 10$

Figure 1 Example of optimal solutions for the SBRP and the SBRP-FT on a benchmark instance.

of the one presented in Erdoğan, Battarra, and Wolfler Calvo (2015) and is built upon an expanded arc structure that enables one to only use binary variables. With this structure, constraints on paths can be used to easily forbid infeasibilities. Our third algorithm removes instead infeasibilities by duplicating vertices associated with stations visited multiple times, but attempts at duplicating as few vertices as possible. Furthermore, we also present some theoretical results on problem complexity

and worst case analysis, as well as an effective feasibility check procedure that is used by all our algorithms.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 presents the aggregate formulation and the branch-and-reject algorithm, whereas Section 4 discusses the complexity of the problem and introduces the theoretical results. Section 5 and 6 describe the binary arc formulation and the minimal extended approach, respectively. Section 7 contains the computational results and Section 8 concludes.

2. Related work

Bike sharing systems are an important tool for supporting sustainable mobility and reducing urban traffic and pollution. Because of this, in recent years there has been a growing trend in self-service bike sharing systems. According to the *Bike Sharing World Map* (see Meddin 2016), in August of 2016 there were at least 1005 cities worldwide with a bike sharing system in operation, and more than 300 others with programs under development. This accounts for, approximately, 1.4 million shared bikes being used on a daily basis.

The SBRP-FT belongs to the class of *pickup and delivery problems* (PDPs), for which we refer to, e.g., the recent surveys by Battarra, Cordeau, and Iori (2014) and Doerner and Salazar-González (2014). According to the classification scheme proposed by Berbeglia et al. (2007), the SBRP-FT is a *many-to-many pickup and delivery problem* (M-M-PDP), in which commodities may have multiple origins and destinations. In the literature, there are a few M-M-PDPs that are related to the SBRP-FT. A notable example is the *one-commodity pickup and delivery traveling salesman problem* (1-PDTSP), introduced by Hernández-Pérez and Salazar-González (2004a), where a single vehicle must perform a series of pickups and deliveries of a single commodity. The 1-PDTSP is different from the SBRP-FT because: (i) all customers must be visited exactly once and thus splits are not allowed; and (ii) the depot serves as a customer and therefore the route is note forced to start and end with an empty vehicle. Hernández-Pérez and Salazar-González (2004a) proposed a mathematical formulation for both the symmetric and the asymmetric versions of the 1-PDTSP, as well as a B&C algorithm. This approach was later improved by Hernández-Pérez and Salazar-González (2007), who proposed an enhanced B&C algorithm strengthened by means of valid inequalities. Heuristic and metaheuristic approaches for the 1-PDTSP were presented by Hernández-Pérez and Salazar-González (2004b), Hernández-Pérez, Rodríguez-Martín, and Salazar-González (2009), Wang, Lim, and Xu (2006), and Mladenović et al. (2012).

Erdoğan, Laporte, and Wolfler Calvo (2014) introduced a variant of the 1-PDTSP, called the *static bicycle relocation problem with demand intervals*, where stations might be visited at most once and the resulting number of bikes at stations after the rebalancing should lie within a given interval

instead of exactly corresponding to a fixed target value. The authors proposed two exact methods to solve the problem and presented valid inequalities. Note that their methods cannot be directly applied to the SBRP-FT because they do not contemplate multiple visits to stations.

Another related problem is the *bike sharing rebalancing problem* (BRP), where a fleet of vehicles is used to serve a set of customers requests, while minimizing the routing costs. In the BRP, stations must be visited exactly once and the vehicles may leave the depot with some initial load. The problem has been studied by Dell'Amico et al. (2014), who presented four mathematical formulations and B&C algorithms. Dell'Amico et al. (2016) later solved the BRP by developing a destroy and repair heuristic, improving the best known solutions for several instances from the literature. Both papers require initially balanced stations to be visited as well, so as to ensure a daily inspection.

A dynamic variant of the BRP was studied by Contardo, Morency, and Rousseau (2012), in which the demand of stations is not static and the fleet of vehicles is heterogeneous. The authors presented a mathematical formulation and proposed a solution approach based on a time-indexed formulation on a fixed time horizon that was solved using Dantzig-Wolfe and Benders' decompositions.

Static rebalancing of bikes was studied also by Raviv, Tzur, and Forma (2013), who aimed at minimizing both operational costs and users' dissatisfaction. The latter term was modeled as a convex function that considers the expected number of shortage events (users cannot rent a bike at an empty station or cannot return a bike at a full station). To solve the problem, they proposed two mathematical formulations, strengthened them with valid inequalities and dominance rules, and conducted tests on a variety of large instances based on real data, showing that small optimality gaps could be achieved within a reasonable time. Recently, Forma, Raviv, and Tzur (2015) solved the same problem by the use of a matching-based heuristic, obtaining good quality solutions for instances with up to 200 stations.

Another static rebalancing variant was addressed by Schuijbroek, Hampshire, and van Hoeve (2017). They combined two types of decisions, namely, determining service level requirements at each bike sharing station and designing low-cost vehicle routes to rebalance the inventory. They solved the resulting decision problem by means of a cluster-first route-second heuristic, in which a polynomial-size clustering subproblem is invoked to simultaneously determine service level feasibility and approximate routing costs. They tested their algorithm on realistic data from the cities of Boston (MA) and Washington (DC).

Recently, Salazar-González and Santos-Hernández (2015) proposed the *split-demand one-commodity pickup and delivery traveling salesman problem* (SD1PDTSP), which is a generalization of the 1-PDTSP. In the SD1PDTSP, the stations and the depot might be visited multiple times by a single vehicle. In practice, the depot is considered as a standard customer having its own demand

and capacity, and the vehicle is not forced to depart from or arrive at the depot empty. In addition, a maximum number of visits is imposed to each station, and split pickups/deliveries as well as temporary operations are allowed. The authors modeled the SD1PDTSP on an extended network where each station is associated to a number of vertices and each vertex is visited at most once. They then developed a B&C algorithm and strengthened it by the use of Benders' decomposition and valid inequalities. They also discussed how to constraint their model to force empty load on vehicles leaving the depot and to forbid temporary operations. Their approach could thus be used to solve the SBRP-FT, although at the expense of the large number of variables required by the underlying extended formulation.

To our knowledge, Chemla, Meunier, and Wolfer Calvo (2013) were the first to study the SBRP, but under the name of single vehicle one-commodity capacitated pickup and delivery problem. The authors proposed a complete formulation to define the problem, presented two relaxations to provide lower bounds, and developed a tabu search algorithm to obtain feasible solutions of good quality. The SBRP was recently heuristically solved by Cruz et al. (2017), who obtained improved results on the existing benchmark instances by means of an iterated local search algorithm.

Erdoğan, Battarra, and Wolfler Calvo (2015) studied the same problem addressed by Chemla, Meunier, and Wolfer Calvo (2013), using the name SBRP, and another problem variant that they called *non-preemptive SBRP*. In the non-preemptive version, a bike can be loaded on the vehicle and unloaded at a station *at most once*, in contrast with the SBRP, where bikes can be temporarily loaded and unloaded any number of times. For the solution of both problem variants they proposed an exact algorithm based on a binary arc formulation (that we resume in Section 5). Note that the non-preemptive SBRP allows temporary operations. Indeed, the situation depicted in Figure 1-(a), although infeasible for the SBRP-FT, is feasible for both the preemptive and the non-preemptive versions of the SBRP studied by Erdoğan, Battarra, and Wolfler Calvo (2015) (the additional bikes picked up and later delivered at vertex 11 are moved only once).

We finally mention the recent work by Bruck and Iori (2017), who presented exact algorithms to deal with the class of one-to-many-to-one single vehicle routing problems with pickups and deliveries, in which a vehicle is used to serve a set of customers requiring a pickup, a delivery, or both. A notable difficulty that is encountered when solving this class of problems is that each customer can be visited up to two times. In Section 6 we propose a non-trivial way of adapting the best algorithm in Bruck and Iori (2017) (minimal extended network algorithm) to the case where demands may have multiple origins and multiple destinations, and stations may be visited more than twice, so as to efficiently deal with the SBRP-FT.

3. Aggregate formulation and branch-and-reject algorithm

In this section, we introduce an effective lower bound that is based on the aggregate formulation by Chemla, Meunier, and Wolfer Calvo (2013), and then discuss a first algorithm that produces an exact SBRP-FT solution.

3.1. Aggregate formulation

As unbalanced stations can be visited an arbitrary number of times, the size of the route may grow exponentially with respect to the number of stations. In the SBRP-FT, as temporary operations are forbidden and the cost matrix satisfies the triangle inequality, a station is visited only to meet at least a part of its demand. In the following, we thus impose that each station $i \in I$ might be visited at most $\beta_i = |d_i|$ times, accounting for the situation in which at each visit a single bike is delivered/collected. Furthermore, under the same assumptions, any arc connecting two stations having both positive or negative demand values is traversed at most once. In this sense, let u_{ij} be an upper bound on the number of traversals on arc $(i,j) \in A$, defined as

$$u_{ij} = \begin{cases} min(|d_i|, |d_j|), & \text{if } (d_i > 0 \text{ and } d_j < 0) \text{ or } (d_i < 0 \text{ and } d_j > 0), \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Let us also define for any set $S \in V$, $\delta^+(S) = \{(i,j) \in A : i \in S, j \in V \setminus S\}$, $\delta^-(S) = \{(i,j) \in A : i \in V \setminus S, j \in S\}$, $\bar{S} = V \setminus S$, and $(\bar{S} : S) = \{(i,j) \in A : i \in \bar{S}, j \in S\}$.

Hereafter, we assume that initially balanced stations are removed from the instances. By setting x_{ij} as an integer variable specifying the number of times arc $(i,j) \in A$ is traversed, we obtain the following *aggregate formulation* (AF):

$$\text{(AF)} \quad \min \; z_{\text{AF}} = \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2}$$

subject to

$$x(\delta^+(0)) = 1 \tag{3}$$

$$x(\delta^+(i)) \geq 1 \qquad \forall \; i \in I \tag{4}$$

$$x(\delta^-(i)) - x(\delta^+(i)) = 0 \qquad \forall \; i \in V \tag{5}$$

$$x(\bar{S} : S) \geq \max\left(\left\lceil \frac{|\sum_{i \in S} d_i|}{Q} \right\rceil, 1 \right) \qquad \forall \; S \subseteq I, \bar{S} = I \setminus S \tag{6}$$

$$x_{ij} \in \{0, 1, .., u_{ij}\} \qquad \forall \; (i,j) \in A \tag{7}$$

The objective function (2) minimizes the total distance traveled. Constraints (3) and (4) define, respectively, the degree for the depot and the minimum degree for the stations, while constraints (5) specify flow conservation for the vehicle. Constraints (6) are classical capacity cut constraints in PDPs (see Hernández-Pérez and Salazar-González 2004a): Any subset S of stations should be visited enough times to ensure deliver or collection of all bikes, and the '1' in the right hand side accounts for the case in which $|\sum_{i \in S} d_i| = 0$. These constraints are separated only at integer nodes

of the branching tree by a twofold procedure. First, we check the existence of subtours using a fast inspection procedure that works by simply traversing the arcs of the solution. For each subtour found, if any, a cut of type (6) is added to the model. In case none are found, we apply a standard max-flow procedure that looks for violations on the vehicle capacity. Constraints (7) define the domain of the variables. In the following, we denote $G = (V, A)$ as the *aggregate network*, and a solution (z, x) to AF as an *aggregate solution*.

3.2. Branch-and-reject algorithm

Formulation AF does not necessarily solve the SBRP-FT to optimality. Instead, it just provides a valid relaxation for the problem. This happens because AF does not consider the evolution on the number of bikes at each visit to a station. As a result, it might produce aggregate solutions with temporary operations, and others where there are no temporary operations but the associated bike displacements are still infeasible. An example of the second case, adapted from a similar example in Chemla, Meunier, and Wolfer Calvo (2013), is shown in Figure 2. In the figure, $I = \{1, 2, 3, 4, 5\}$, each depicted arc is traversed just once, and the vehicle load is reported over each arc. It is easy to see that, for $Q \geq 2$, this aggregate solution satisfies all constraints (3)–(7), and is thus feasible for AF. However, it is infeasible for the SBRP-FT, because when the vehicle arrives at station 2, starting in 0 and passing through 1, it has only 1 bike to deliver as station 5 has not yet been visited. It is worth mentioning that this kind of infeasibilities may only exist in aggregate solutions and when there is at least one vertex visited more than once.

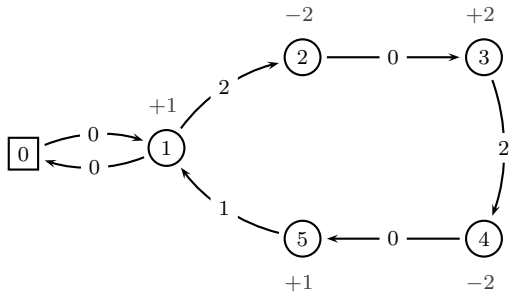


Figure 2 Example of an aggregate solution that is feasible for AF but infeasible for the SBRP-FT.

Because the x variables are integer (not just binary), it is not possible to separate the standard *infeasible path constraints*, nor the enhanced *combinatorial Benders' cuts* (see, e.g., Codato and Fischetti 2006) to forbid infeasible aggregate solutions.

A straightforward approach to overcome this issue and define a complete problem formulation could be achieved by solving the SBRP-FT under a so-called *extended network*, obtained by duplicating

each station $i \in I$ into β_i vertices, and imposing each vertex to be visited at most once. By doing so, we could keep track of the evolution of the number of bikes at each station during the successive visits, thus building *extended solutions* where the aforementioned infeasibilites for the SBRP-FT are eliminated. This was indeed the approach implemented by Salazar-González and Santos-Hernández (2015) to solve the SD1PDTSP, and one of the approaches implemented by Bruck and Iori (2017) to solve the *single vehicle routing problem with deliveries and selective pickups* (a one-to-many-to-one PDP in which deliveries are mandatory, pickups are optional but generate a revenue if performed, and customers are visited at most twice). However, these two works showed that formulations on the extended network become easily intractable, even for small size instances, due to the large number of duplicate vertices and the even larger number of possible connections between the vertices. Indeed, the maximum number of visits to a vertex is 2 in the tests in Bruck and Iori (2017), and 3 in the tests in Salazar-González and Santos-Hernández (2015). We thus disregarded this approach from our tests as typical β_i values of our instances can be much larger than 2 or 3.

Our algorithms rely instead on the idea of building feasible SBRP-FT solutions starting from feasible AF solutions (or from reformulations/generalizations of AF). To this aim, following the same notation adopted for the SBRP by Chemla, Meunier, and Wolfer Calvo (2013), we say that an aggregate solution $(\bar{z}, \bar{x})$ to AF *induces* a feasible SBRP-FT solution if it is possible to find a route that travels exactly $\bar{x}_{ij}$ times on each arch (i, j), has cost $\bar{z}$, and satisfies all SBRP-FT constraints. The following feasibility check is solved a number of times by our algorithms.

Definition 1. Given an aggregate solution $(\bar{z}, \bar{x})$ to AF, the *disaggregation check* is to determine whether or not $(\bar{z}, \bar{x})$ induces a feasible solution for the SBRP-FT.

In the following, we call *disaggregate solution* a feasible SBRP-FT solution that is produced by a disaggregation check. Our first solution approach, denoted as *branch-and-reject* (B&R) algorithm, consists in solving AF under the B&C framework provided by a *mixed integer linear programming* (MILP) solver, but solving the disaggregation check for any integer solution that is encountered during the process. AF solutions inducing feasible disaggregate solutions are kept and used to possibly update the incumbent solution. AF solutions for which the disaggregation check returned answer "no" are instead reported to be infeasible and hence disregarded. More in detail, there are two possible sources for an AF infeasible solution: either (i) it has been generated by the general purpose heuristics of the MILP solver, or (ii) it originates from a current integral node of the search tree. In case (i) the solution is simply rejected, whereas in case (ii) the corresponding node is fathomed without adding explicit constraints. Note that, when an integer point is declared infeasible, the branching process within a MILP solver continues, and thus B&R implicitly explores all integer points in the polyhedron and returns an optimal solution.

Details on the method that we have implemented to solve the disaggregation check are given in Section 6.3 and the B&R is computationally evaluated in Section 7.

4. Complexity and worst case analysis

In this section, we discuss the computational complexity of the disaggregation check and study the worst-case performance of some important PDP problems.

4.1. Complexity of the disaggregation check

We prove the complexity of the disaggregation check by means of two consecutive results. For the sake of conciseness, all proofs of this paper are included in the appendix.

PROPERTY 1. The disaggregation check is $\mathcal{NP}$-complete.

Proof. In the appendix.

This first result derives from a similar one that has been proposed for the SBRP by Chemla, Meunier, and Wolfer Calvo (2013), but it is based on a slightly different proof that allows to better understand our second and stronger result.

PROPERTY 2. The disaggregation check is strongly $\mathcal{NP}$-complete.

Proof. In the appendix.

4.2. Worst case analysis

In the well-known *vehicle routing problem* (VRP), a fleet of vehicles must visit a set of customers, each exactly once, so as to satisfy demands and minimize costs. Archetti, Savelsbergh, and Speranza (2006) studied the relaxation in which each customer can be visited more than once (a.k.a. *split-delivery VRP*), thus allowing demand splits. In particular, they showed that $z(\text{VRP})/z(\text{VRP with splits}) \leq 2$, where $z(\cdot)$ denotes the optimal solution value, and that the bound is tight. This proves that allowing splits may halve the VRP solution cost.

A related study was conducted by Nowak, Ergun, and White (2008) on the area of *one-to-one* PDPs (refer again to the notation by Berbeglia et al. 2007). The authors focused on the *pickup and delivery VRP* (PDVRP), a VRP generalization in which each customer demand consists of transporting a load from a given pickup vertex to a given destination vertex, and conjectured that $z(\text{PDVRP})/z(\text{PDVRP with splits}) \leq 2$. To the best of our knowledge this conjecture still holds.

We investigated similar properties in the area of M-M PDPs, where, in contrast to the VRP and the PDVRP, demands may have multiple origins and destinations. We could not find a relationship between the SBRP (that allows splits and temporary operations) and the SBRP-FT (that only allows splits), neither between the SBRP and the SBRP variant in which only splits are forbidden. However, we can point out the relationship that exists between the SBRP and the SBRP variant in which *both* splits and temporary operations are forbidden, and show that the same result applies to the SD1PDTSP studied by Salazar-González and Santos-Hernández (2015).

PROPERTY 3. Forbidding both the demand splitting and the use of temporary operations in the SBRP may lead to an arbitrarily large increase in the solution cost, and the same holds for the SD1PDTSP.

Proof. In the appendix.

It is worth mentioning that, as often happens on combinatorial problems, there is quite a large gap between a proven worst case and what happens on average in practical cases. In fact, Cruz et al. (2017) have empirically shown that the difference in the solution cost, when one forbids temporary operations in the SBRP, is relatively small. They conducted experiments with and without temporary operations, and found that the increase in the solution cost was always smaller than 4% in their tests.

5. Binary arc formulation

Erdoğan, Battarra, and Wolfler Calvo (2015) proposed an exact algorithm to solve the SBRP and further adapted it to the non-preemptive SBRP. In this section we present a direct extension of their algorithm which is capable of solving the SBRP-FT.

As mentioned in Section 3, the main drawback that prevents AF from exactly solving the SBRP-FT is the fact of not considering the evolution of the number of bikes at stations after each visit. Provided that the triangle inequality holds, we can use (1) to transform each integer variable x_{ij} in AF into a set of binary variables as

$$x_{ij} = \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ij}^k \qquad \forall\ (i,j) \in A \tag{8}$$

where y_{ij}^k is a binary variable assuming value 1 if arc (i,j) is traversed at least k times by the vehicle. Equations (8) can be intuitively seen as a representation of the value of x_{ij} as a binary number, where each variable y_{ij}^k gives the k-th bit of the number (see, e.g., Vanderbeck and Wolsey 2010). For the sake of simplicity, let $\gamma_{ij} = \lfloor log_2(u_{ij}) \rfloor$. We now create a multigraph in which each arc $(i,j) \in A$ is replaced by a set of arcs (i,j,k) with $k = 0, 1, \ldots, \gamma_{ij}$. Let $\mathcal{R}$ be the set of routes that are *infeasible* for the SBRP-FT and $A'(r)$ be the subset of arcs in the multigraph used by route $r \in \mathcal{R}$. We are now ready to present the following *binary arc formulation* (BinArc).

$$\text{(BinArc)} \quad \min\ z_{\text{BinArc}} = \sum_{(i,j)\in A} \sum_{k=0}^{\gamma_{ij}} c_{ij} 2^k y_{ij}^k \tag{9}$$

subject to

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{i0}} 2^k y_{i0}^k = 1 \tag{10}$$

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \geq 1 \qquad \forall\ j \in I \tag{11}$$

$$\sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \leq u_{ij} \qquad \forall\ (i,j) \in A \tag{12}$$

$$\sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k - \sum_{i \in V} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ji}^k = 0 \qquad \forall\ j \in V \tag{13}$$

$$\sum_{i \in \bar{S}} \sum_{j \in S} \sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k \geq \max\left(\left\lceil \frac{|\sum_{i \in S} d_i|}{Q} \right\rceil, 1 \right) \qquad \forall\ S \subseteq I \tag{14}$$

$$\sum_{(i,j,k)\in A'(r)} y_{ij}^k \leq |A'(r)| - 1 \qquad \forall\ r \in \mathcal{R} \qquad (15)$$

$$y_{ij}^k \in \{0,1\} \qquad \forall\ (i,j) \in A, k = \{0,\ldots,\gamma_{ij}\} \qquad (16)$$

The objective function (9) minimizes the total distance traveled. Constraints (10)-(12) impose the degrees for the depot and the stations. Constraints (13) specify that each arrival at a vertex is followed by a departure, while (14) are capacity constraints for the vehicle. Infeasible routes are forbidden by constraints (15), whereas constraints (16) define the domain of the y variables. Note that constraints (12) are not strictly necessary, but are used to strengthen the formulation, given that $\sum_{k=0}^{\gamma_{ij}} 2^k y_{ij}^k$ might be greater than u_{ij}.

In practice, the model made by (9)–(14) and (16) is a reformulation of AF, and thus a valid relaxation, but the inclusion of (15), made possible by the use of the binary expansion, allows to obtain a complete SBRP-FT representation. To separate (14), we first use equation (8) to aggregate the y variables, and then we apply a standard max-flow procedure. To separate (15), we solve the disaggregation check using the method described later in Section 6.3. In short, formulation (9)-(16) is solved by a B&C procedure, called BinArc algorithm, that separates constraints (14) and (15) only at integer nodes of the branching tree.

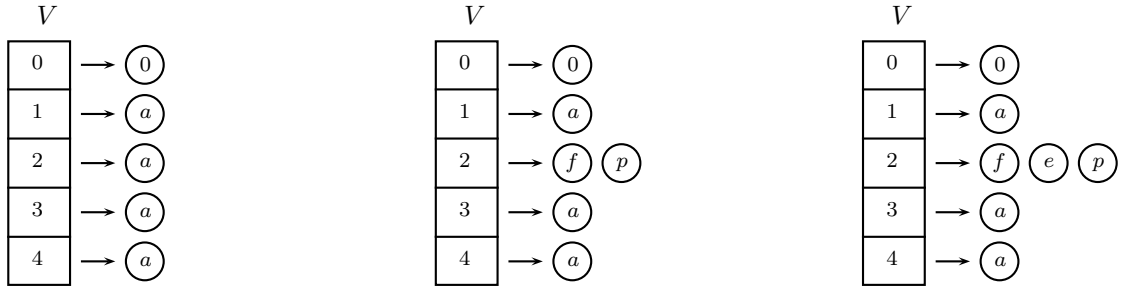
6. The minimal extended network approach

In this section, we introduce a formulation capable of modeling any state of the problem network, from the aggregate network, where each station is associated with a single vertex, to the extended network, where each station i is associated with β_i vertices. We then use the formulation as a basis of an iterative B&C algorithm.

6.1. General formulation

Let $G'' = (V'', A'')$ be a complete and directed graph, where $V'' = \cup_{i\in I} V_i \cup \{0\}$ is the set of vertices, $V_i = \{i_1,\ldots,i_{\bar{\beta}_i}\}$ is the subset of vertices associated with station $i \in I$, and $\bar{\beta}_i$ is a parameter satisfying $\bar{\beta}_i \leq \beta_i$ for each $i \in I$. In addition, let I_k specify the station associated with vertex $k \in V''$. To represent the evolution that our iterative algorithm imposes on this network, set V'' is partitioned as $V'' = V^a \cup V^p \cup V^f \cup V^e \cup \{0\}$. As an illustrative example consider the vertex set with depot and 4 stations depicted in Figure 3-(a). This set contains only *aggregate vertices* that belong to V^a and can be visited multiple times. Suppose that vertex 2 is duplicated by our iterative procedure, as shown in Figure 3-(b). The initial aggregate vertex is replaced by a *first extended vertex* belonging to V^f, and a *partially aggregate vertex* belonging to V^p. Vertices in V^f are visited at most once, while those in V^p may be visited multiple times. Also, a partially aggregate vertex may only be visited when all other vertices associated with the same station are also visited. Suppose now that a second duplication of vertex 2 occurs, as shown in Figure 3-(c). This results in an *extended vertex*, belonging to V^e, which

may be visited at most once. To better understand the difference between V^e and V^f, consider two vertices, $k \in V^f$ and $l \in V^e$, having $I_k = I_l = i \in I$. The only difference between k and l is that the former can only be visited when the latter is also visited. Additionally, in case $d_i \neq 0$, vertex k must be visited exactly once. Subsequent duplications of the same vertex, not explicitly reported in the figure, would maintain exactly one vertex of type f and one vertex of type p, and increase only the number of vertices of type e.



(a) Initial state of the vertex set (b) First duplication of vertex 2 (c) Second duplication of vertex 2

Figure 3 Example of a vertex duplication

Similarly to AF, let x_{ij} be a set of integer variables specifying the number of times arc (i,j) in A'' is traversed, and let y_i be a set of binary variables that define whether or not vertex $i \in V'' \setminus \{0\}$ is visited. In addition, let g_i be an unconstrained variable that reports the quantity of demand that has been collected/delivered during the visit to vertex $i \in V'' \setminus \{0\}$. We are now ready to introduce the following *general formulation* (GF), which is the stepping stone of our algorithm:

$$\text{(GF)} \quad \min \; z_{\text{GF}} = \sum_{(i,j)\in A''} c_{ij} x_{ij} \tag{17}$$

subject to

$$x(\delta^+(0)) = 1 \tag{18}$$

$$x(\delta^+(j)) = 1 \qquad \forall\, j \in V^f \tag{19}$$

$$x(\delta^+(j)) \geq 1 \qquad \forall\, j \in V^a \tag{20}$$

$$x(\delta^+(j)) \geq y_j \qquad \forall\, j \in V^p \tag{21}$$

$$x(\delta^+(j)) = y_j \qquad \forall\, j \in V^e \tag{22}$$

$$x(\delta^+(j)) - x(\delta^-(j)) = 0 \qquad \forall\, j \in V'' \tag{23}$$

$$y_{i_k} \geq y_{i_{k+1}} \qquad \forall\, i \in I, k \in \{1, \ldots, \bar{\beta}_i - 1\} \tag{24}$$

$$\sum_{k \in V_i} g_k = d_i \qquad \forall\, i \in I \tag{25}$$

$$x(\bar{S} : S) \geq \max\left(\frac{-\sum_{i\in S} g_i}{Q}, 1\right) \qquad \forall\, S \subseteq V'' \setminus \{0\}, \bar{S} = V'' \setminus S \setminus \{0\}, S \cap (V^f \cup V^a) \neq \emptyset \tag{26}$$

$$x(\bar{S}:S) \geq \frac{-\sum_{i\in S} g_i}{Q} \qquad \forall\ S \subseteq V'' \setminus \{0\}, \bar{S} = V'' \setminus S \setminus \{0\}, S \cap (V^f \cup V^a) = \emptyset \qquad (27)$$

$$y_i \leq g_i \leq d_{I_i} y_i \qquad \forall\ i \in V'' \setminus \{0\} : d_{I_i} \geq 0 \qquad (28)$$

$$d_{I_i} y_i \leq g_i \leq -y_i \qquad \forall\ i \in V'' \setminus \{0\} : d_{I_i} < 0 \qquad (29)$$

$$x_{ij} \in \{0,1\} \qquad \forall\ i,j \in V'' : i \in V^e \cup V^f \text{ or } j \in V^e \cup V^f \qquad (30)$$

$$x_{ij} \in \{0,1,\ldots,u_{I_i I_j}\} \qquad \forall\ i,j \in V^a \cup V^p \qquad (31)$$

$$y_i \in \{0,1\} \qquad \forall\ i \in V'' \setminus \{0\} \qquad (32)$$

$$g_i \gtreqless 0 \qquad \forall\ i \in V'' \setminus \{0\} \qquad (33)$$

The objective function (17) minimizes the total distance traveled. Constraints (18)-(22) define the degree of each vertex. Constraints (23) specify flow conservation for the vehicle for each vertex in V''. Constraints (24) impose the order for the usage of the duplicates, stating that a vertex i_{k+1} can only be visited if its predecessor i_k is also visited. The demand d_i of each station must be served, as stated by constraints (25), and the vehicle capacity Q must not be exceeded, as imposed by (26) and (27). Note that constraints (26) and (27) also serve the purpose of preventing subtours and are separated using the same procedure applied to constraints (6). The *max* function in the right hand side of (26) induces a nonlinear term, which is required because $-\sum_{i\in S} g_i$ could be 0 or lower. We deal with this by simply checking at each call to the separation procedure which term is the highest, and then using it as the right hand side of the constraint that is actually added to the model. Constraints (28) and (29) specify lower and upper bounds for the g_i variables, which should be equal to 0 when $y_i = 0$, lower than 0 when $y_i = 1$ and $d_{I_i} < 0$, and greater than 0 when $y_i = 1$ and $d_{I_i} < 0$. It is worth mentioning that constraints similar to (28)–(29) have been proposed by Salazar-González and Santos-Hernández (2015) to model the SD1PDTSP. Constraints (30)-(33) define the variables domains, with g_i being a continuous variable unrestricted in sign.

This formulation has some interesting properties. In case each station is associated with a single (aggregate) vertex, we have $V'' = V^a \cup \{0\}$ and $V^p = V^f = V^e = \emptyset$, which corresponds to the scenario adopted for AF. Moreover, $g_j = d_{I_j}$ for each $j \in V''$, and consequently constraints (26) and (27) are equivalent to (6), and constraints (28) and (29) can be removed, as they become redundant. This leads to:

Observation 1. In case each station is associated with a single vertex in G'', GF is equivalent to AF.

Now recall that the shortcomings that might render an aggregate solution infeasible may only exist when there is at least one vertex being visited multiple times. By associating each station $i \in I$ with β_i duplicates, we obtain the extended network, where each vertex is visited at most once. Therefore, solving GF under this network always produces a feasible solution, and hence:

Observation 2. In case each station $i \in I$ is associated with β_i duplicates in G'', an optimal solution of GF is also an optimal solution for SBRP-FT.

6.2. MEN algorithm

Our third exact algorithm, called the *minimal extended network* (MEN) algorithm, generalizes the best algorithm in Bruck and Iori (2017) to deal with the case where demands have multiple origins and destinations and stations may be visited more than twice. The MEN algorithm is based on the following two observations. Firstly, most often than not, in optimal solutions for the SBRP-FT stations are visited no more than 2 or 3 times (this behavior was already hinted by Salazar-González and Santos-Hernández (2015) for the related SD1PDTSP). Secondly, by performing extensive computational experiments, we have verified that quite often an optimal solution of AF corresponds to an optimal solution for the SBRP-FT.

Algorithm MEN starts by solving GF over the pure aggregate network, where each station is associated with a single vertex. Formulation GF is solved by using a B&C algorithm in which constraints (26) and (27) are separated with the max-flow procedure described in Chemla, Meunier, and Wolfer Calvo (2013). We opted to separate those constraints only at integer nodes. A disaggregation check is then performed on the solution obtained by the B&C to determine whether the solution is feasible for the SBRP-FT (details are provided in Section 6.3). If feasible, then MEN terminates with an optimal SBRP-FT solution. Otherwise, we inspect the solution and determine the number $\bar{\beta}_i$ of visits to i. Each vertex $i \in V''$ having $\bar{\beta}_i > 1$ is duplicated into $\bar{\beta}_i$ vertices. Then, a new iteration solves GF under the modified graph. The procedure iterates until the optimal solution found is feasible for the SBRP-FT.

Recall that AF provides a valid relaxation of the SBRP-FT. Because of this fact and of Observation 1, algorithm MEN provides at the first iteration a valid lower bound. Then, in the worst case scenario, it iterates for $\sum_{i \in I} \beta_i$ times, performing a single duplication at each iteration, possibly improving the lower bound value, and culminating in the extended network in the last iteration, where, because of Observation 2, it is guaranteed to produce a feasible SBRP-FT solution. Consequently, the solution value would be both a lower and an upper bound, and hence an optimum, so we can conclude that:

Observation 3. Algorithm MEN converges to the optimal solution of the SBRP-FT.

According to Observation 3, algorithm MEN might be as time consuming (or even more) as directly solving the extended network. However, computational experiments indicate this is not the case and, in practice, only a few iterations are usually necessary to converge to an optimal SBRP-FT solution. Furthermore, to speed up convergence, at each iteration of MEN a list of the 10 best incumbent solutions found is kept. In case the solution found is infeasible, this list is checked as an attempt of finding a feasible upper bound. The incumbent upper bound is then used as a warm start for the next iteration. During computational experiments we observed that, for some instances, the first iteration of MEN produces an infeasible aggregate solution $\bar{x}$, but the reverse solution $\tilde{x}$, having $\tilde{x}_{ij} = \bar{x}_{ji}$ for

all $(i,j) \in A''$, is actually feasible. To take advantage of this observation, an additional feasibility test is performed in the reverse solution of the first MEN iteration, if needed.

It is worth mentioning that an extended network approach was already used by Salazar-González and Santos-Hernández (2015) to model the related SD1PDTSP, so the most fruitful contribution of algorithm MEN consists in iteratively solving the problem, starting from the simplest aggregate network and then iteratively extending the graph through additional vertices until an optimum is found. To this regard, note that there is no guarantee that after the execution of algorithm MEN the resulting graph will be of overall minimum cardinality. In fact, finding a graph of minimal cardinality that could allow MEN to produce an optimal solution appears to be as hard as solving the SBRP-FT itself (and also finding non-trivial bounds on such cardinality is not easy). Even duplicating just one vertex at a time, instead of duplicating all vertices i with $\bar{\beta}_i > 1$, has no guarantee to lead to a minimum cardinality.

It is also worth mentioning that the MEN algorithm can be easily extended to deal with the case of rebalancing by means of a fleet of multiple vehicles. This can be obtained by simply replacing "1" with the number of available vehicles in constraint (18). Note, indeed, that because we forbid temporary operations there is no transshipment, and this in turn prevents synchronization issues that could require additional constraints.

6.3. A duplicate-and-inspect procedure to solve the disaggregation check

Let $(\bar{z}, \bar{x}, \bar{g})$ be a solution of GF of value $\bar{z}$, and let $\bar{V}^{>1} = \{i : i \in V'', \bar{\beta}_i > 1\}$ and $\bar{V}^1 = V'' \setminus \bar{V}^{>1}$. In addition, define $\tilde{G} = (\tilde{V}, \tilde{A})$ as the support graph obtained by duplicating each vertex $i \in \bar{V}^{>1}$ into $\bar{\beta}_i$ duplicates, and including all vertices in $\bar{V}^1$ directly into $\tilde{V}$. Let $\sigma(i)$ specify the vertex in V'' that is associated with $i \in \tilde{V}$. We check whether or not $(\bar{z}, \bar{x}, \bar{g})$ leads to a disaggregate feasible solution for the SBRP-FT by solving a formulation, called *duplicate and inspect* (DIF) that uses the same variables x and g as in GF, but is defined over $\tilde{G}$ as follows:

$$\text{(DIF)} \quad \min\ 0 \tag{34}$$

subject to

$$(23), (25), (26), (27)$$

$$\sum_{i \in \tilde{V}} \sum_{j \in \tilde{V}} c_{ij} x_{ij} = \bar{z} \tag{35}$$

$$\sum_{i \in \tilde{V}} x_{ij} = \sum_{i \in \tilde{V}} \bar{x}_{\sigma(i)\sigma(j)} \qquad \forall\ j \in \bar{V}^1 \tag{36}$$

$$\sum_{i \in \tilde{V}} x_{ij} = 1 \qquad \forall\ j \in \tilde{V} \setminus \bar{V}^1 \tag{37}$$

$$x_{ij} = \bar{x}_{\sigma(i)\sigma(j)} \qquad \forall\ i,j \in \bar{V}^1 \tag{38}$$

$$x_{ij} = 0 \qquad \forall\ i,j \in \tilde{V} : \{\sigma(i), \sigma(j)\} \cap \bar{V}^{>1} \neq \emptyset, \bar{x}_{\sigma(i)\sigma(j)} = 0 \tag{39}$$

$$0 \leq g_i \leq d_{\sigma(i)} \qquad \forall\ i \in \tilde{V} : d_{\sigma(i)} \geq 0 \qquad (40)$$

$$-d_{\sigma(i)} \leq g_i \leq 0 \qquad \forall\ i \in \tilde{V} : d_{\sigma(i)} < 0 \qquad (41)$$

$$x_{ij} \in \{0,1\} \qquad \forall\ i,j \in \tilde{V} \qquad (42)$$

$$g_i \lessgtr 0 \qquad \forall\ i \in \tilde{V} \qquad (43)$$

Constraints (35) ensure that the search is performed for a disaggregate solution of the same value $\bar{z}$ of the original aggregate one, whereas constraints (36) and (37) impose that stations must be visited the same number of times as in the original solution. Variables connecting vertices in $\bar{V}^1$ keep the same values as in the solution of GF, as stated by constraints (38). Based on the $\bar{x}$ values, constraints (39) remove several arcs that should not be used. Consequently, because in this formulation vertices are visited at most once, constraints (40) and (41) are enough to forbid temporary operations. In case formulation DIF returns a feasible solution, then this is also an SBRP-FT optimal solution and the associated route may be retrieved by simply inspecting the values assumed by the x variables.

The disaggregation check can also be used to analyze solutions from the B&R of Section 3.2 and the BinArc of Section 5. In both cases, because each station is represented by a single vertex, the $\bar{g}$ values are simply obtained by setting $\bar{g}_i = d_i$ for each $i \in I$. For BinArc, the $\bar{x}$ values are computed by using equation (8).

7. Computational experiments

Our algorithms have been coded in C++ and the computational experiments have been run on a PC with an Intel Core i7-3770 3.40 GHz with 8 GB of RAM. A time limit of 1 hour has been given to each execution. We have used CPLEX 12.6.2 with default options and single thread to solve the formulations and to implement the B&C algorithms.

We first tested our algorithms on a set of instances obtained by slightly modifying those proposed by Chemla, Meunier, and Wolfer Calvo (2013) for the SBRP. The results on these instances are summarized in Section 7.1. We then selected our two best solution methods and used them to solve two additional sets: a set of realistic instances proposed by Dell'Amico et al. (2014) on the basis of a large analysis on a number of real-world bike sharing systems; and a set of particularly difficult instances that we derived by a real-world consideration from Liu et al. (2016). The results on these two test beds are reported in Section 7.2. Details of the results that we obtained on each single instance appear in the appendix.

The proposed duplicate-and-inspect procedure runs quite fast on average. Nevertheless, when there are several stations being visited multiple times, its performance tends to degrade. As a matter of fact, because the MEN algorithm runs the feasibility check only at the end of each iteration and the solutions checked are usually of high quality (thus containing few occurrences of multiple visits

to a station), there were no significant efficiency issues with the duplicate-and-inspect procedure. However, this was not the case for both the B&R and the BinArc algorithms, as the check is called every time an incumbent solution is found, and the first solutions found are likely to be of poor quality (possibly containing several stations that are visited multiple times). We overcame this issue by providing an initial solution of good quality. In view of this, we have adapted to the SBRP-FT the *iterated local search* (ILS) algorithm proposed by Cruz et al. (2017) for the SBRP, and used it to provide an initial solution (called UB_0) for both the B&R and the BinArc algorithm. This has been obtained by using the MIPstart procedure provided by CPLEX. The two resulting algorithms are called B&R+MIPstart and BinArc+MIPstart below. In addition, we tested two variants for MEN: a first one, called MEN+MIPstart, in which UB_0 is used as a MIPstart when solving the GF models; and a second one, simply called MEN, in which UB_0 is only used to provide an output solution in case MEN did not find any lower cost solution.

7.1. Computational results on SBRP benchmark instances

We have slightly modified the set of benchmark instances proposed by Chemla, Meunier, and Wolfer Calvo (2013) for the SBRP and later used by Erdoğan, Battarra, and Wolfler Calvo (2015), by imposing triangle inequality on the cost matrix (original costs were obtained by rounding down to the nearest integer euclidean distances, and this can create small violations of the triangle inequality). To this aim, we performed a simple iterative check that, for each triplet (i, j, k) of vertices, updates $c_{ij} = c_{ik} + c_{kj}$ whenever $c_{ik} + c_{kj} < c_{ij}$, and reiterates until no further update exists. We have obtained in this way 450 instances divided into 10 classes, with a number of stations n varying from 20 to 60, and a vehicle capacity Q varying from 10 to 1000. The demand of each station is an integer number randomly chosen with uniform distribution in the range $[-10, 10]$. All the instances that we tested are now available at `http://www.or.unimore.it/site/home/online-resources.html`.

Due to the large number of instances, in this section we only present aggregate results, but refer to the appendix for detailed results on each particular instance. The appendix also contains detailed information on n, Q, and demand distribution for each instance. For each algorithm in the tables presented hereafter, and for each group of # instances in a line, *opt* specifies the total number of optimal solutions found, *gap* provides the average percentage gap obtained by the respective algorithm for the instance, and *sec* gives the average CPU time in seconds (considering 1 hour for instances unsolved to proven optimality). The gap values are evaluated as ((UB-LB)/UB)×100), where UB and LB are the upper and lower bound values produced by the given algorithm (UB=UB_0 if no improvement was found on the initial solution provided by the ILS). The best opt values are highlighted in bold. For UB_0 (that is, the ILS of Cruz et al. 2017) we provide in column UB_{opt} the

number of times in which the provided upper bound value was equal to the proven optimal value, and in column sec_0 the average CPU time. The last line reports total values for UB_{opt} and opt and average values for gap and times across the entire set of instances.

Table 1 Aggregated results per instance size.

		UB_0		BinArc+MIPstart			B&R+MIPstart			MEN+MIPstart			MEN		
n	#	UB_{opt}	sec_0	opt	gap	sec	opt	gap	sec	opt	gap	sec	opt	gap	sec
20	90	89	7.06	84	0.24	250.36	86	0.07	162.06	**90**	0.00	2.73	**90**	0.00	4.31
30	90	83	24.50	71	0.46	869.47	**90**	0.00	11.31	88	0.15	81.91	89	0.05	41.57
40	90	80	77.64	42	1.00	2092.86	81	0.25	614.11	83	0.47	375.42	**85**	0.26	281.07
50	90	64	160.98	37	2.64	2220.91	66	1.29	1116.18	70	1.43	859.27	**72**	0.84	797.15
60	90	66	251.46	25	3.98	2613.17	53	2.69	1701.22	63	2.70	1175.04	**69**	1.82	962.69
total/avg	450	382	104.33	259	1.66	1609.36	376	0.86	720.98	394	0.95	498.87	**405**	0.59	417.36

Table 1 reports aggregate results per instance size. We can observe that the BinArc algorithm is usually outperformed by B&R and MEN, as it found only 259 optimal solutions, whereas B&R proved the optimality of 376 instances, and MEN with and without MIPstart managed to find 394 and 405 optimal solutions, respectively. In addition, the BinArc formulation required, on average, more CPU time. Furthermore, the MEN algorithm has a better overall performance as the size of the instance increases.

Two interesting comments can be made when comparing MEN+MIPstart with MEN. On one side, it is interesting to observe that providing an initial solution for the MEN algorithm did not improve its overall efficiency. In fact, for several instances the opposite behavior was verified. This might be explained by the observation that GF usually finds feasible upper bounds quite fast, but takes longer to improve the lower bound, and providing a warm start might change the branching decisions taken by CPLEX. The fact that performing modifications in the initial conditions of a MILP solver may lead to significant changes in the resulting performance has already been noticed by Fischetti and Monaci (2014) and Lodi and Tramontani (2013), among others. On the other side, there are instances for which MEN alone cannot provide any feasible solution. That happened for 2 instances with $n = 40$, 5 instances with $n = 50$, and 10 instances with $n = 60$ (we refer to the appendix for details). In these cases, the use of a good initial heuristic is very important.

Table 2 shows the aggregate results per vehicle capacity. Once again MEN presents a better performance, usually dominating the other algorithms in terms of number of optimal solutions, gap, and CPU time. One exception occurred for $Q = 10$, where B&R found 26 optimal solutions against 24 of MEN. From Table 2 it can be also verified that the larger the capacity, the easier the instance. In fact, instances having Q larger than 35 are not very meaningful, as they are easily solved by both

Table 2 Aggregated results per vehicle capacity.

		UB_0		BinArc+MIPstart			B&R+MIPstart			MEN+MIPstart			MEN		
Q	#	UB_{opt}	sec_0	opt	gap	sec	opt	gap	sec	opt	gap	sec	opt	gap	sec
10	50	21	122.50	14	7.45	2672.03	**26**	5.17	1830.23	23	5.81	2052.99	25	3.78	1842.37
15	50	32	118.45	23	2.78	1932.07	34	1.59	1198.13	32	1.90	1324.73	**38**	1.22	974.71
20	50	38	109.40	28	1.12	1675.96	41	0.43	760.39	42	0.67	705.37	**44**	0.29	589.17
25	50	46	104.31	31	0.61	1397.53	48	0.07	362.22	**49**	0.06	185.51	**49**	0.00	110.16
30	50	47	99.62	35	0.53	1188.45	48	0.02	286.59	**49**	0.09	106.57	**49**	0.04	156.48
35	50	48	94.74	34	0.57	1268.70	46	0.10	404.36	49	0.04	81.92	**50**	0.00	49.58
40	50	50	97.34	32	0.53	1451.79	45	0.07	516.47	**50**	0.00	6.48	**50**	0.00	6.45
45	50	50	97.12	32	0.65	1378.32	44	0.13	554.08	**50**	0.00	16.46	**50**	0.00	15.13
1000	50	50	95.49	30	0.73	1519.34	44	0.15	576.31	**50**	0.00	9.84	**50**	0.00	12.18
total/avg	450	382	104.33	259	1.66	1609.36	376	0.86	720.98	394	0.95	498.87	**405**	0.59	417.36

MEN+MIPstart and MEN. In addition, it is visible that all algorithms have a quite inferior overall performance for $Q = 10$ and $Q = 15$, mainly because the number of multiple visits increases as the capacity of the vehicle decreases, and this negatively affects the efficiency of the exact algorithms presented in this work.

7.2. Computational results on realistic benchmark instances

We also tested our best algorithms on a set of real-world instances that were first proposed by Dell'Amico et al. (2014). These instances are derived from a large analysis on a number of real-world bike sharing systems, involving between 13 and 116 stations. Dell'Amico et al. (2014) studied the case in which multiple homogeneous vehicles can be used and each station is visited once. We adapted the instances to the SBRP-FT by assuming that a single vehicle is used, removing initially balanced stations, and allowing the unbalanced stations to be visited more than once. In addition, to deal with the fact that, originally, these instances are not balanced (i.e., $\sum_{i \in I} d_i \neq 0$), we added a dummy station at the same coordinates of the depot and set its demand as $-\sum_{i \in I} d_i$. The vehicle capacity was kept as in Dell'Amico et al. (2014), who attempted up to 3 different values for each graph. Note that we limited our study to the 56 instances having at most 80 stations, because this is the computational limit of our exact methods and also because it is unrealistic to consider that a single vehicle can visit more stations in a single day.

Table 3 presents aggregate results for UB_0, MEN+MIPstart, and MEN, by grouping instances according to the original graph (i.e., city). Detailed results for each instance are provided in the appendix. In general, MEN still performs well, proving 43 out of 56 optimal solutions against 41 of MEN+MIPstart. However, in 7 cases MEN was not able to find a feasible solution, while MEN+UB_0 always guaranteed that at least one is found. It is interesting to point out that the instances associated

with Buenos Aires have a small number of stations, but seem harder to solve than some larger instances. This is probably due to the fact that the demands of stations in Buenos Aires are very high, and the vehicle capacity is very tight, and this makes the problems more difficult to solve. On the other hand, instances associated with Guadalajara have a larger number of stations but very low demand values, and are easier to solve. In Table 4 we aggregate the results by value of Q. Also here one can notice that small vehicle capacities tend to increase the solution difficulty. We also performed an additional test by attempting Q=1000, and could solve all instances with both MEN+MIPstart and MEN in at most a few minutes.

Table 3 Aggregate results per city for the real-world instances.

			UB_0		MEN+MIPstart			MEN		
city	n	#	UB_{opt}	sec_0	opt	gap	sec	opt	gap	sec
Bari	13	3	3	0.39	**3**	0.00	0.01	**3**	0.00	0.01
Reggio Emilia	14	3	2	0.65	**3**	0.00	0.03	**3**	0.00	0.05
Bergamo	15	3	3	0.65	**3**	0.00	0.02	**3**	0.00	0.08
Parma	15	3	3	0.54	**3**	0.00	0.02	**3**	0.00	0.02
Treviso	18	3	3	0.85	**3**	0.00	0.04	**3**	0.00	0.02
La Spezia	20	3	0	1.00	**3**	0.00	0.31	**3**	0.00	0.34
Buenos Aires	21	2	0	2.13	0	0.72	t.lim.	**1**	7.67	1802.24
Ottawa	21	3	3	1.13	**3**	0.00	0.20	**3**	0.00	0.34
San Antonio	23	3	0	2.10	**3**	0.00	0.11	**3**	0.00	0.10
Brescia	27	3	0	3.53	**3**	0.00	7.08	**3**	0.00	4.71
Roma	28	3	0	4.44	**2**	0.13	1206.20	**2**	1.40	1203.90
Madison	28	3	1	2.99	**2**	0.33	2401.25	**2**	0.62	2135.80
Guadalajara	41	3	0	7.81	**3**	0.00	0.92	**3**	0.00	3.11
Dublin	45	3	0	11.41	**1**	1.74	2403.12	**1**	1.70	2403.15
Denver	51	3	0	18.42	**3**	0.00	118.96	**3**	0.00	124.79
Rio de Janeiro	55	3	0	21.69	0	4.56	t.lim.	0	14.00	t.lim.
Boston	59	3	0	31.84	**2**	0.56	2251.36	**2**	6.17	1755.28
Torino	75	3	0	54.32	**2**	6.47	1894.57	**2**	12.59	1576.50
Toronto	80	3	0	55.12	0	33.84	t.lim.	0	36.83	t.lim.
total/avg		56	18	11.80	42	2.58	1065.22	**43**	4.20	943.38

Table 4 Aggregate results per vehicle capacity for the real-world instances.

		UB_0		MEN+MIPstart			MEN		
Q	#	UB_{opt}	sec_0	opt	gap	sec	opt	gap	sec
[10,20)	18	5	9.93	**13**	3.17	1185.50	**13**	5.67	1104.26
20	19	7	10.78	13	2.79	1248.02	**14**	5.27	1156.63
30	19	6	14.60	15	1.80	768.48	**16**	1.74	577.71
total/avg	56	18	11.80	42	2.58	1065.22	**43**	4.20	943.38

We also decided to perform additional tests that follow the observation by Liu et al. (2016), who pointed out that in many practical applications the demand distribution is usually geographically

unbalanced. For example, as people usually leave their residencies in the morning, residential areas are expected to have a shortage of bikes in this moment of the day, while commercial regions have a surplus. Similarly, in the late afternoon this scenario is usually reversed. To tackle this type of scenarios, we created a new set of instances where the demand of stations is distributed unevenly. These instances were generated by modifying the demands of the instances proposed by Chemla, Meunier, and Wolfer Calvo (2013) using the following procedure. For each instance, we first divide the geographic space into four quadrants as in the two-dimensional Cartesian system. Next, stations in quadrants 1 and 3 are assigned a large pickup demand equal to a random value in the interval $[\lceil Q/2 \rceil, \lfloor 3Q/2 \rfloor]$, whereas stations in quadrants 2 and 4 are assigned a large delivery demand equal to a random value in the interval $[\lceil -3Q/2 \rceil, \lceil -Q/2 \rceil]$. We selected the instances of classes A to E, with $n = \{20, 30, 40, 50, 60\}$ and $Q = 25$ (the same capacity value used in Liu et al. 2016), resulting in a new set of 25 instances with realistic spatial distribution of demand. Note that the procedure we adopted tends to generate demands with values that are much higher than those of the other instances that we tested. The resulting instances are expected to be very challenging because split deliveries are forced by the fact that some demands are larger than the vehicle capacity.

The outcome of the computational experiments performed on this new set of instances is presented in Table 5. Detailed results are reported in the appendix. As expected, the instances are more challenging to solve. This can be noted by the reduced quality of the initial upper bound provided by the heuristic, which now never manages to reach the value of a proven optimal solution, and by the reduced scalability of the MEN algorithm, which now can solve to optimality only instances having 40 stations or less. The positive aspect is that the developed algorithms can directly tackle instances with demands that are higher than the vehicle capacity, showing a good adaptability to a wide range of problems.

Table 5 Aggregated results per instance size for instances with realistic spatial demand distribution.

		UB_0		MEN+MIPstart			MEN		
n	#	UB_{opt}	sec_0	opt	gap	sec	opt	gap	sec
20	5	0	0.06	5	0.00	0.56	**5**	0.00	0.53
30	5	0	0.22	4	7.45	1581.93	**4**	4.19	99.09
40	5	0	0.48	1	9.07	2984.77	**2**	14.15	2711.78
50	5	0	0.87	0	21.93	t.lim.	0	18.25	t.lim.
60	5	0	1.54	0	28.93	t.lim.	0	21.51	t.lim.
total/avg	25	0	0.64	10	13.48	2353.45	**11**	13.12	2002.28

8. Conclusions

In this paper, we have introduced the static bike rebalancing problem with forbidden temporary operations (SBRP-FT), a pickup and delivery problem that arises in bike rebalancing, courier service transportation, and repositioning of inventory cases in which temporary dropoff/pickup of products is forbidden. We have presented some theoretical results related to computational complexity and worst case analysis and proposed three exact algorithms. The first one consists of a branch-and-reject (B&R) algorithm that builds upon an incomplete integer programming formulation. The second one is an integer programming model, denoted as BinArc formulation, that is based on a binary network expansion. The third one is based on the so-called minimum extended network (MEN) algorithm, which iteratively enlarges a relaxed formulation by duplicating as few vertices as possible.

We have first performed extensive computational experiments with the proposed exact methods on a set of 450 benchmark instances. Although the BinArc formulation is usually outperformed by the other two approaches, it is capable of proving the optimality of 278 solutions, while B&R and MEN (using its best configuration) obtain 376 and 402 optimal solutions, respectively. The good behavior of the MEN algorithm was also proven by a second set of tests that we performed on realistic instances taken from the bike sharing literature.

Future work includes the application of the MEN algorithm to other routing problems with split demands, and the development of hybrid algorithms by combining heuristic and exact approaches to find improvements for the SBRP-FT open instances. Such instances are mainly those with small capacity values and in practice they tend to be harder to solve because the feasible solutions usually contain several multiple visits to the stations.

Acknowledgments

The authors acknowledge financial support by the Brazilian research agencies CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) under grant PVE no. A007/2013 and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), under grant no. 305223/2015-1, and by the Italian ministry MIUR under grant PRIN 2015. We finally thank two anonymous referees, whose comments helped improve the quality of the paper.

References

Archetti C, Savelsbergh M, Speranza M, 2006 *Worst-Case Analysis for Split Delivery Vehicle Routing Problems. Transportation Science* 40(2):226–234.

Battarra M, Cordeau JF, Iori M, 2014 *Pickup and delivery problems for goods transportation.* Toth P, Vigo D, eds., *Vehicle Routing: Problems, Methods, and Applications*, 161–192, MOS-SIAM Series on Optimization (SIAM), 2nd edition.

Berbeglia G, Cordeau JF, Gribkovskaia I, Laporte G, 2007 *Static pickup and delivery problems: aÂĂclassification scheme and survey. Top* 15(1):1–31.

Bruck B, Iori M, 2017 *Non-elementary formulations for single vehicle routing problems with pickups and deliveries. Operations Research* 65:1597–1614.

Cefic–The European Chemical Industry Council, 2013 *Best practice guidelines for safe (un)loading of road freight vehicles.* available at `http://www.cefic.org` (last accessed December 2017).

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

Codato G, Fischetti M, 2006 *Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. Operations Research* 54(4):756–766.

Contardo C, Morency C, Rousseau LM, 2012 *Balancing a dynamic public bike-sharing system.* Technical Report CIRRELT-2012-09, Universitè de Montrèal, Montrèal, Canada.

Cruz F, Bruck B, Subramanian A, Iori M, 2017 *A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. Computers & Operations Research* 79:19–33.

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

Dell'Amico M, Iori M, Novellani S, Stützle T, 2016 *A Destroy and Repair Algorithm for the Bike sharing Rebalancing Problem. Computers & Operations Research* 71:149–162.

Doerner K, Salazar-González JJ, 2014 *Pickup and delivery routing problems for people transportation.* Toth P, Vigo D, eds., *Vehicle Routing: Problems, Methods, and Applications*, 193–212, MOS-SIAM Series on Optimization (SIAM), 2nd edition.

Erdoğan G, Battarra M, Wolfler Calvo R, 2015 *An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. European Journal of Operational Research* 245(3):667–679.

Erdoğan G, Laporte G, Wolfler Calvo R, 2014 *The static bicycle relocation problem with demand intervals. European Journal of Operational Research* 238(2):451–457.

Fischetti M, Monaci M, 2014 *Exploiting Erraticism in Search. Operations Research* 62(1):114–122.

Forma I, Raviv T, Tzur M, 2015 *A 3-step math heuristic for the static repositioning problem in bike-sharing systems. Transportation Research Part B: Methodological* 71:230–247.

Hernández-Pérez H, Rodríguez-Martín I, Salazar-González JJ, 2009 *A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. Computers & Operations Research* 36(5):1639–1645.

Hernández-Pérez H, Salazar-González JJ, 2004a *A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics* 145(1):126–139.

Hernández-Pérez H, Salazar-González JJ, 2004b *Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem. Transportation Science* 38(2):245–255.

Hernández-Pérez H, Salazar-González JJ, 2007 *The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. Networks* 50(4):258–272.

Liu J, Sun L, Chen W, Xiong H, 2016 *Rebalancing bike sharing systems: A multi-source data smart optimization. Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1005–1014, KDD '16 (New York, NY, USA: ACM).

Lodi A, Tramontani A, 2013 *Performance Variability in Mixed-Integer Programming*, chapter 1, 1–12. TutORials in Operations Research (INFORMS).

Meddin R, 2016 *The bike-sharing world map.* `http://www.bikesharingmap.com`, accessed: April, 2018.

Mladenović N, Urošević D, Hanafi S, Ilić A, 2012 *A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. European Journal of Operational Research* 220(1):270–285.

Nowak M, Ergun O, White C, 2008 *Pickup and Delivery with Split Loads. Transportation Science* 42(1):32–43.

Official Journal of the European Union, 2013 *Guidelines of 7 march 2013 on good distribution practice of medicinal products for human use (2013/c 68/01).* available at `http://ec.europa.eu/` (last accessed December 2017).

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

Salazar-González JJ, Santos-Hernández B, 2015 *The split-demand one-commodity pickup-and-delivery travelling salesman problem. Transportation Research Part B: Methodological* 75:58–73.

Schuijbroek J, Hampshire R, van Hoeve WJ, 2017 *Inventory rebalancing and vehicle routing in bike sharing systems. European Journal of Operational Research* 257(3):992 – 1004.

Vanderbeck F, Wolsey L, 2010 *Reformulation and decomposition of integer programs.* Jünger M, Liebling T, Naddef D, Nemhauser G, Pulleyblank W, Reinelt G, Rinaldi G, Wolsey L, eds., *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, 431–502 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Wang F, Lim A, Xu Z, 2006 *The one-commodity pickup and delivery travelling salesman problem on a path or a tree. Networks* 48(1):24–35.