The bike sharing rebalancing problem: Mathematical formulations and benchmark instances

Article *in* Omega · January 2013

DOI: 10.1016/j.omega.2013.12.001

4 authors:

Mauro Dell'Amico
Università degli Studi di Modena e Reggio Emilia
83 PUBLICATIONS **2,507** CITATIONS
SEE PROFILE

Eleni Hadjiconstantinou
Frederick University Nicosia, Cyprus
32 PUBLICATIONS **1,087** CITATIONS
SEE PROFILE

Manuel Iori
Università degli Studi di Modena e Reggio Emilia
65 PUBLICATIONS **1,648** CITATIONS
SEE PROFILE

Stefano Novellani
Università degli Studi di Modena e Reggio Emilia
6 PUBLICATIONS **67** CITATIONS
SEE PROFILE

Omega 45 (2014) 7–19

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

The bike sharing rebalancing problem: Mathematical formulations and benchmark instances

Mauro Dell'Amico [a], Eleni Hadjicostantinou [b,c], Manuel Iori [a,*], Stefano Novellani [a,d]

[a] DISMI, University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia, Italy
[b] Business School, Imperial College London, South Kensington Campus, London SW7 2AZ, UK
[c] School of Engineering, Frederick University, 7 Y. Frederickou Str., 1036 Nicosia, Cyprus
[d] DEI – "Guglielmo Marconi", University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

ARTICLE INFO

Article history:
Received 17 April 2013
Accepted 9 December 2013
Processed by B. Lev
Available online 18 December 2013

Keywords:
Integer programming
Routing
Traveling salesman
Vehicle scheduling

ABSTRACT

Bike sharing systems offer a mobility service whereby public bicycles, located at different stations across an urban area, are available for shared use. These systems contribute towards obtaining a more sustainable mobility and decreasing traffic and pollution caused by car transportation. Since the first bike sharing system was installed in Amsterdam in 1965, the number of such applications has increased remarkably so that hundreds of systems are now operating all over the world.

In a bike sharing system, users can take a bicycle from a station, use it to perform a journey and then leave it at a station, not necessarily the same one of departure. This behavior typically leads to a situation in which some stations become full and others are empty. Hence, a balanced system requires the redistribution of bicycles among stations.

In this paper, we address the Bike sharing Rebalancing Problem (BRP), in which a fleet of capacitated vehicles is employed in order to re-distribute the bikes with the objective of minimizing total cost. This can be viewed as a special one-commodity pickup-and-delivery capacitated vehicle routing problem. We present four mixed integer linear programming formulations of this problem. It is worth noting that the proposed formulations include an exponential number of constraints, hence, tailor-made branch-and-cut algorithms are developed in order to solve them.

The mathematical formulations of the BRP were first computationally tested using data obtained for the city of Reggio Emilia, Italy. Our computational study was then extended to include bike sharing systems from other parts of the world. The information derived from the study was used to build a set of benchmark instances for the BRP which we made publicly available on the web. Extensive experimentation of the branch-and-cut algorithms presented in this paper was carried out and an interesting computational comparison of the proposed mathematical formulations is reported. Finally, several insights on the computational difficulty of the problem are highlighted.

1. Introduction

Bike sharing systems offer a mobility service in which public bicycles are available for shared use. These bicycles are located at stations that are displayed across an urban area. The users of the system can take a bicycle from a station, use it for a journey, leave it into a station (not necessarily the one of departure), and then pay according to the time of usage.

These systems are an important instrument used by public administrations to obtain a more sustainable mobility, decrease traffic and pollution caused by car transportation, and solve the so-called last mile problem related to proximity travels. From the first bike sharing system installed in Amsterdam in 1965, their number increased in the following years to reach, in 2011, more than 400 systems only in Europe, see, e.g., DeMaio [1] and project OBIS [2]. In North America the implementation of bike sharing systems started only in 2008, see Pucher et al. [3], but as far as we know it already counts more than 20 operating systems. In the rest of the world the number of systems is rising at a very high rate, as discussed, e.g., by Shaheen et al. [4].

Stations are made of different slots, each of which hosts a single bicycle. In modern systems, stations are connected to the Internet and display in real time the occupation status of each slot. In this way users can easily check where it is possible to pick up or drop a bicycle. The usage of the system is monitored continuously, and the collected information is used to improve the level of service.

Operating bike sharing systems has a cost that may vary greatly (depending on the system itself, the population density, the service area and the fleet size), with a consistent impact on the budget of the public administration. The setup costs for installing the system

* Corresponding author. Tel.: +39 0522 522653.
E-mail addresses: mauro.dellamico@unimore.it (M. Dell'Amico), e.hconstantinou@imperial.ac.uk (E. Hadjicostantinou), manuel.iori@unimore.it (M. Iori), stefano.novellani2@unibo.it (S. Novellani).

0305-0483/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.omega.2013.12.001

include, among others, the cost of purchasing the bikes, the slots, and the stations, and the cost of the back-end system used to operate the equipment, see, e.g., DeMaio [1]. The daily operating costs include maintenance, insurance, possibly website hosting and electricity, and, most important, the cost due to the redistribution of bikes among the stations. Indeed, at the end of a day some stations are typically full and others are empty.

A commonly adopted rule for rebalancing is to keep each station only partly occupied, i.e., there should always be in a station some slots occupied by bicycles, to allow users to pick them up, and some free slots, to allow users to drop a bicycle at the end of their journey. Let us suppose that a desired level of occupation is present in the early morning in a given bike station, then the number of bikes may change drastically during the day from the desired level because of the users' travel behavior. This happens typically in cities characterized by a hilly territory, see, e.g., Kaltenbrunner et al. [5], where users take a bike from a station located at the top of a hill, leave it at the bottom and then take the journey back with different means of transportation. It is also common for cities located in flat areas, where some stations have large inflows or outflows at different times of the day. In the next section we report the results of the analysis of the system at the city of Reggio Emilia (Italy) used over a period of seven months.

Repositioning is usually done by means of capacitated vehicles based on a central depot that pick up bicycles from stations where the level of occupation is too high and deliver them to stations where the level is too low. Usually a buffer of bicycles is kept at the depot, and used to allow a more flexible redistribution. The resulting optimization problem of deciding how to route the vehicles so as to perform the redistribution at minimum cost is known in the literature as the *Bike sharing Rebalancing Problem* (BRP), and has recently attracted the interest of many researchers and practitioners in the area. It can be modeled as either a *dynamic* or a *static* optimization problem. In the static version, a snapshot of the level of occupation at the stations is taken and then used to plan the redistribution. In the dynamic version, the real-time usage of the system is taken into account, and the redistribution plan is possibly updated as soon as the information required to make decisions is revealed over time.

Usually, static rebalancing is associated with a redistribution process that is performed during the night, when the system is kept closed or the demand is very low, whereas dynamic rebalancing is associated to redistributions operated during the day, when demand may be high. In the real-world case that we studied in detail the redistribution is performed during the night, and hence we focus on the static version of the problem.

In this paper we provide several contributions. In Section 2 we briefly present the real-world case study that we conducted at the city of Reggio Emilia, by analyzing the travel flows, the users behavior and the resulting levels of occupation at the stations. In Section 3 we formally describe the BRP and discuss the related literature. In Section 4 we propose four *Mixed Integer Linear Programming* (MILP) formulations to model the problem. All these formulations involve an exponential number of constraints, so Section 5 presents the branch-and-cut algorithms that we implemented to solve them. We present a large set of benchmark instances in Section 6, obtained by analyzing the usage of several bike systems around the world; we make these instances publicly available on the Internet. Extensive computational results of the branch-and-cut algorithms are reported in Section 7. Finally, conclusions are given in Section 8 and future research directions are discussed.

2. Data analysis of a real-world case

The first real-world case that we studied is the bike sharing system of Reggio Emilia, a city of around 170 thousand inhabitants located in a very flat area in northern Italy. The system, which is depicted in Fig. 1, is quite small and now counts one depot (indicated by 0 in the figure), 13 stations and about 100 bicycles. It operates all day but is kept closed during the night, which is quite common for bike sharing systems in small/medium cities. The redistribution of the bicycles is carried out during the night, by means of a single vehicle that visits each station exactly once.

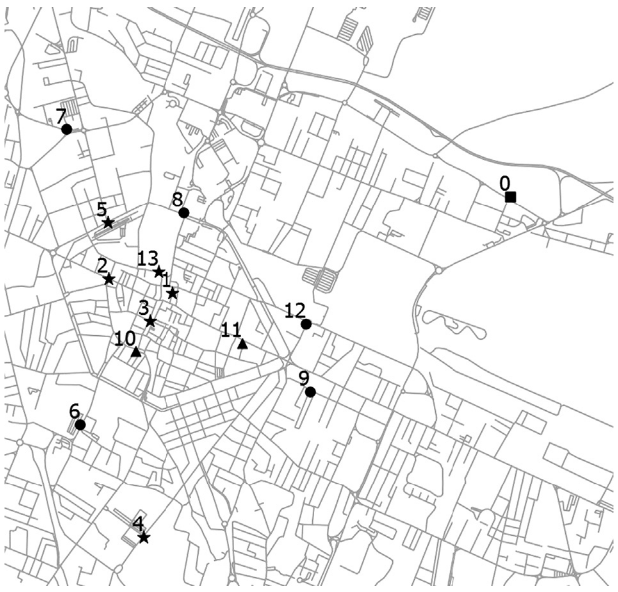

Fig. 1. The bike sharing system of Reggio Emilia. The depot is depicted by 0. Stars, circles and triangles represent stations of the first, second and third groups, respectively.

The data associated with a seven-month usage of the system was provided to us by the municipality of the city. It contains the list of journeys performed by the users in the considered period, including time and station of departure and arrival of each journey. For each station we evaluated the net flow of bicycles on a daily basis, computed as the difference between the inflow and the outflow. This gives the difference between the bicycles available at the beginning of the day and those left at the end of the day in each station. We then plotted the distribution of the net flow over the period, see the distribution graphs in Fig. 2. The x-axis gives the difference Δ between arrivals and departures in a station per day, and the y-axis gives the percentage of times Φ (frequency of occurrence) this number appears throughout the period that we studied. We mostly found normal-like distributions, as the one depicted in Fig. 2(a) for station 4, but also a bimodal one for station 5, see Fig. 2(b). In both cases, for the majority of the days over the observation the stations ended up with a number of bikes different from that available at the beginning of the day, and this supports the choice of performing rebalancing operations.

Furthermore, by analyzing the net flow per hour of all the stations we have been able to determine the diverse variability in usage by the customers. We could consequently divide the stations into three groups, as shown in Fig. 3. In this figure, the x-axis represents the hour τ of the day, and the y-axis states the cumulative number ν of bikes arriving into or departing from the station within a seven-months period. More specifically:

1. The first group, see Fig. 3(a), has a peak of incoming bikes between 7 and 9 am, and a smaller one between 1 and 3 pm. The peaks of outgoing bikes occur between 12 am and 2 pm and between 4 and 6 pm. These stations are all situated in the city center, with the exception of one that is located near the hospital (stations 1, 2, 3, 4, 5 and 13). This usage fits well with

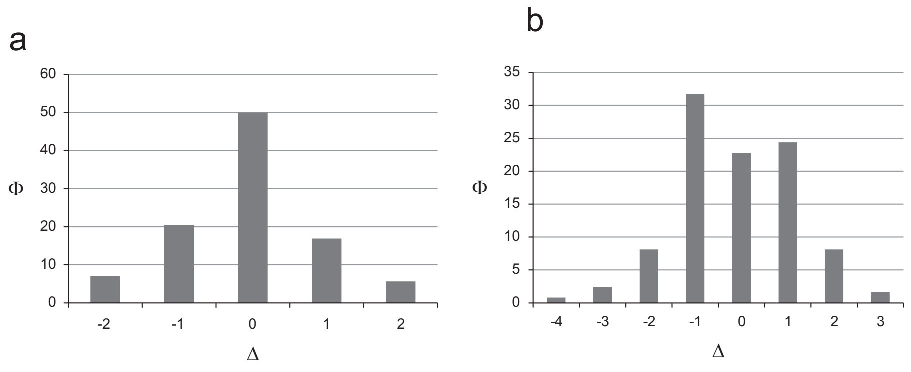

Fig. 2. Net flow of bicycles per day, following (a) a normal-like distribution in station 4 and (b) a bimodal distribution in station 5 (Δ, difference between arrivals and departures; Φ, frequency of occurrence).

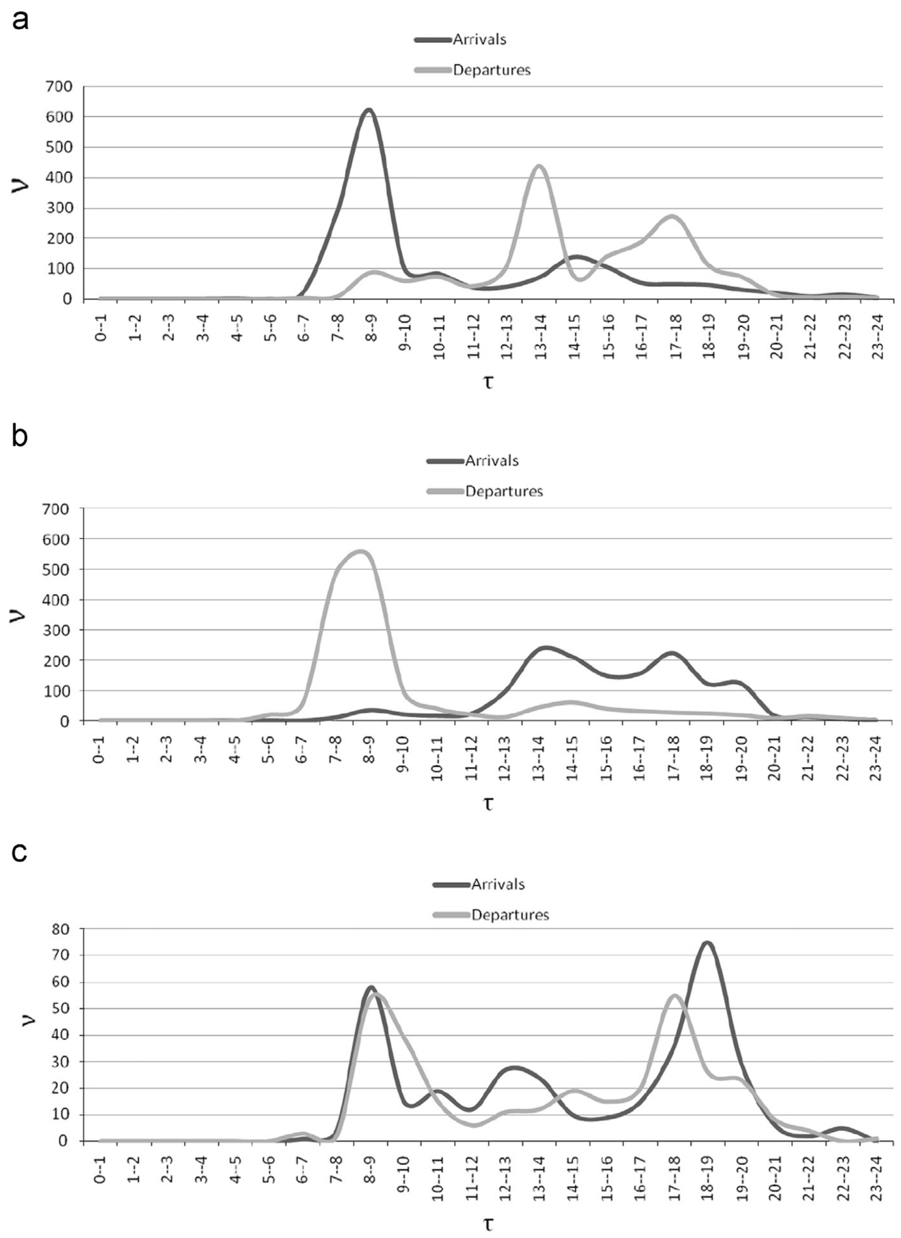

Fig. 3. Typical arrivals and departures per hour in a station of (a) the first group, (b) the second group and (c) the third group (τ, hour of the day; ν, cumulative number of bikes arriving into or departing from a station within a seven-months period).

the behavior of users that work in the city center but live outside.
2. The second group is complementary to the first one, see Fig. 3(b). These stations (6, 7, 8, 9 and 12) are in the so-called park-and-ride areas, where users can leave their cars and continue their ride on a public bike.
3. The third group, see Fig. 3(c), contains stations where arrivals and departures follow a similar pattern during the day. These stations (10 and 11) are located close to places, such as the train station, that are used all day long.

More enhanced studies on bike traveling habits are out of the scope of this paper, so the interested reader is referred to the relevant literature. Vogel and Mattfeld [6] present a business model for bike sharing systems. They model the repositioning activities with the help of a system dynamics approach that they solve with a simulation tool. They conclude that "more effort spent on repositioning leads to a better corporate performance in terms of satisfied customers". Kaltenbrunner et al. [5] provide an analysis of human mobility data in the urban area of Barcelona (Spain) using the amount of available bikes in the stations of the local bike sharing program. By using data sampled from the operator's website, they detect temporal and geographic mobility patterns within the city. Lin and Yang [7] address the strategic planning of public bicycle sharing systems. These authors propose a model that attempts to determine the number and locations of bike stations, the users' travel paths, and the network structure of bike trips connecting the stations.

The determination of the most suitable bike trips has been studied by Souffriau et al. [8], for a problem arising in East Flanders (Belgium), and motivated by the problem that a cyclist deals with when looking for a nice route of a certain length. They propose a mathematical model and a metaheuristic. The metaheuristic obtained good computational results and was then embedded into a web-based cycle route planner. We also mention that a similar problem was faced in a different context by Boctor et al. [9], who studied the assignment of trips to vehicles in petrol stations replenishment problems.

3. Problem description and previous work

We are given a complete digraph $G=(V,A)$, where the set of vertices $V=\{0,1,\ldots,n\}$ is partitioned into the depot, vertex 0, and the stations, vertices $\{1,2,\ldots,n\}$. Each station i has a request q_i, which can be either positive or negative. If $q_i \ge 0$ then i is a *pickup* node, where q_i bikes should be removed; if $q_i < 0$ then i is a *delivery* node, where q_i bikes should be supplied, for $i \in V \setminus \{0\}$. The bikes removed from pickup nodes can either go to a delivery node or back to the depot. Bikes supplied to delivery nodes can either come from the depot or from pickup nodes. A fleet of m identical vehicles of capacity Q is available at the depot to transport the bikes. A traveling cost c_{ij} is associated with each arc $(i,j) \in A$.

The BRP involves determining how to drive at most m vehicles through the graph, with the aim of minimizing the total cost and ensuring that the following constraints are not violated: (i) each vehicle performs a route that starts and ends at the depot, (ii) each vehicle starts from the depot empty or with some initial load (i.e., with a number of bikes that vary from 0 to Q), (iii) each station is visited exactly once and its request is completely fulfilled by the vehicle visiting it, and (iv) the sum of requests of the visited stations plus the initial load is never negative or greater than Q in the route performed by a vehicle.

In our study, each request q_i is computed as the difference between the number of bikes present at station i when performing the redistribution, and the number of bikes in the station in the final required configuration. Note that we impose that a station with request $q_i=0$ must be visited, even if this implies that no bike has to be dropped off or picked up there. This case arises, for example, when the driver of the vehicle is supposed to check that the station is correctly working. The case in which stations with null requests have to be skipped can be simply obtained by removing in a preprocessing phase those stations from the set of vertices.

Table 1
BRP notation summary.

V	Set of vertices
A	Set of arcs
n	Number of stations
m	Number of vehicles
Q	Vehicle capacity
q_i	Demand at vertex i
c_{ij}	Cost of the arc (i,j)

The fact that each vehicle is allowed to start its route with some bikes enlarges the space of feasible BRP solutions, and allows to obtain a more flexible redistribution plan. Note also that we do not impose the sum of redistributed bikes to be null, and hence there can be a positive or a negative flow of bikes on the depot. This consideration is useful to model cases in which some bikes enter or leave the depot for maintenance.

The traveling cost c_{ij} is computed in our case as the shortest length of a path in the road network connecting i and j, for $i,j \in V$. It is important to work on a directed graph, because all bike sharing systems we are aware of are located in urban areas, and thus one-way streets typically have a strong impact on the choice of the routes performed by the vehicles during the redistribution.

The notation defined for the BRP is summarized in Table 1.

The BRP belongs to the wide class of *Pickup and Delivery Vehicle Routing Problems* (PDVRPs), where a fleet of vehicles is used to transport requests from the depot and/or some nodes to the depot and/or other nodes in the network. In particular, some *pickup customers* require a certain amount of freight to be picked up by a vehicle and then transported elsewhere, whereas some *delivery customers* require a certain amount of freight to be delivered there. The BRP generalizes the well-known *Capacitated Vehicle Routing Problem* (CVRP), where customers are either all pickup vertices or all delivery vertices, but not both, and thus it is a strongly NP-hard problem.

Detailed reviews and classifications of the PDVRPs have been proposed by Berbeglia et al. [10] and Parragh et al. [11,12]. More recent surveys have been presented by Battarra et al. [13], for what concerns the transportation of freight, and Doerner and Salazar-González [14], for the transportation of people. Following the notation introduced in Berbeglia et al. [10], the BRP can be classified as a *Many-to-Many* (M–M) vehicle routing problem, in which a request has multiple origins (in our case pickup stations) and multiple destinations (in our case delivery stations).

The most basic problem in the class of M–M PDVRPs is probably the *One-commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP), which calls for routing a single capacitated vehicle to meet M–M requests. The 1-PDTSP was formally introduced by Hernández-Pérez and Salazar-González [15], who presented mathematical formulations and branch-and-cut algorithms. The mathematical formulations were given for both the symmetric and asymmetric versions of the problem, and were based on the use of a binary variable x_{ij} taking value one if the edge or arc (i,j) was used, 0 otherwise, and a non-negative variable f_{ij} giving the flow of the single commodity on edge or arc (i,j). Computational results were provided only for the symmetric formulation that was solved by means of a branch-and-cut algorithm. The results of this approach were later improved by

Hernández-Pérez and Salazar-González [16], with an enhanced branch-and-cut algorithm working again on the symmetric formulation but exploiting only the binary variable x_{ij}. To address larger instances, Hernández-Pérez and Salazar-González [17] proposed two simple heuristics, whereas a well-performing variable neighborhood descent metaheuristic was introduced by Hernández-Pérez et al. [18]. All these works considered the case in which vehicles can leave the depot with some load, but the sum of all requests is equal to zero.

The 1-PDTSP has also attracted the interests of other researchers. Wang et al. [19] studied some variants of the 1-PDTSP that involve unit-load requests. They developed polynomial-time exact algorithms for the case of distribution on a path, and for the cases of distribution on a tree having $Q=1$ or $Q=+\infty$. Martinovic et al. [20] presented a simulated annealing approach, whereas Zhao et al. [21] developed a genetic algorithm. The results of these two metaheuristics were later improved by Hosny and Mumford [22] by means of a *Variable Neighborhood Search* (VNS), but at the expense of very high computing times. A more recent VNS has been proposed by Mladenović et al. [23] that used a binary tree to efficiently perform feasibility checks and obtained very good computational results.

The multiple-vehicle case, known as the *One-commodity Pickup and Delivery Vehicle Routing Problem* (1-PDVRP), was formally introduced by Shi et al. [24], who studied the case in which a maximum duration limit is imposed on each route. They presented a genetic algorithm and a three-index mathematical formulation making use of a binary variable x_{ijk}, taking value 1 if vehicle k travels along edge or arc (i,j), 0 otherwise, and a non-negative variable f_{ij} representing the flow of the single commodity on edge or arc (i,j). Computational experiments were performed only for the genetic algorithm that was tested on a set of randomly generated instances with symmetric costs.

The BRP is a generalization of the 1-PDTSP which involves more than one vehicle. Indeed, in the next sections we use successful properties and algorithmic ideas adapted for the BRP from [15,16]. Moreover, the BRP can be seen as a special case of the 1-PDVRP, because it does not consider maximum route duration.

As previously mentioned, the issue of rebalancing a bike sharing system has attracted the interest of many researchers in recent years. Benchimol et al. [25] study the case in which the redistribution is performed by a single capacitated vehicle, split deliveries are allowed (i.e., a vertex may be visited more than once) and the sum of all requests is balanced to zero. They present complexity results, lower bounding techniques, and approximation algorithms, but do not provide computational results.

In a recent paper, Chemla et al. [26] focus again on the single vehicle case, allowing split deliveries. Vertices with null demand are not forced to be visited, but can be used by vehicles as temporary buffer when transporting bikes among other stations. In their paper, the authors obtain a lower bound by solving the relaxation of the problem in which a limit is imposed on the maximum number of times that a vertex can be visited. They also obtain an upper bound by using a tabu search algorithm and run their bounding procedures on instances obtained by modifying those randomly created in [15].

In another recent paper, Raviv et al. [27] define the *static bicycle repositioning problem* in which they do not only minimize total traveling costs of a fleet of heterogeneous vehicles, but also users' dissatisfaction that is linked with a convex function to the inventory level of each station. The authors present two MILP formulations, an arc-indexed and a time-indexed one, each of them represents a different version of the problem. They allow limited or unlimited split delivery, limited or unlimited transshipment, dwelling with a maximum duration of each route and with synchronized transshipment. Exact and heuristics methods to speed up the computational times for solving the formulations are also presented in this paper.

Contardo et al. [28] present a formulation for the dynamic version of the bike rebalancing problem, allowing the use of more than one vehicle. Their objective function is the maximization of the serviced demand. An arc flow formulation working on a discretized time horizon is proposed and solved with Dantzig–Wolfe and Benders decomposition. Extensive tests on several randomly created instances are presented.

4. MILP formulations for the BRP

In this section we present four *Mixed Integer Linear Programming* (MILP) formulations of the BRP. Computational evidence obtained from preliminary experiments suggested to us that it would be better to adopt two-index formulations for the BRP.

4.1. Formulation F1

The starting point of the aforementioned two-index formulations is the well-known *Multiple Traveling Salesman Problem* (*m*-TSP), see, e.g., Bektas [29], in which at most m uncapacitated vehicles based on a central depot have to visit a set of vertices, with the constraint that each vertex is visited exactly once. By defining a binary variable x_{ij}, taking value 1 if arc (i,j) is used by a vehicle, 0 otherwise, the m-TSP can be modeled as

$$(m\text{–TSP}) \ \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

$$\sum_{i \in V} x_{ij} = 1, \quad j \in V \backslash \{0\} \tag{2}$$

$$\sum_{i \in V} x_{ji} = 1, \quad j \in V \backslash \{0\} \tag{3}$$

$$\sum_{j \in V} x_{0j} \le m \tag{4}$$

$$\sum_{j \in V \backslash \{0\}} x_{0j} = \sum_{j \in V \backslash \{0\}} x_{j0} \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \quad S \subseteq V \backslash \{0\}, \ S \neq \varnothing \tag{6}$$

$$x_{ij} \in \{0,1\}, \quad i,j \in V. \tag{7}$$

Objective function (1) minimizes the traveling cost. Constraints (2) and (3) impose that every node but the depot is visited exactly once. Constraints (4) and (5) ensure, respectively, that at most m vehicles leave the depot, and that all vehicles that are used return to the depot at the end of their route. Constraints (6) are the classical *subtour elimination constraints*, see, e.g., Gutin and Punnen [30], that impose the connectivity of the solution. These constraints increase exponentially, so, as usual, we proceed in branch-and-cut fashion, by invoking a separation procedure to identify the violated ones from the non-violated ones, and then adding to the model only the violated constraints in an iterative way. The details will be given in the next section.

To guarantee the feasibility of a solution with respect to the BRP, we need to include additional constraints in the m-TSP formulation, so as to ensure that demands are satisfied and vehicle capacities are not exceeded. This may be done in different ways.

The first option is to define an additional continuous variable θ_j, representing the load of a vehicle after it visited node j, for $j \in V$. The load should be updated along the route by taking into consideration the fact that, if the vehicle travels along arc (i,j), then θ_j should be equal to $\theta_i + q_j$. The aforementioned restriction can be imposed by the non-linear constraints:

$$\theta_j \geq (\theta_i + q_j)x_{ij}, \quad i \in V, \ j \in V \backslash \{0\} \tag{8}$$

$$\theta_i \geq (\theta_j - q_j)x_{ij}, \quad i \in V \backslash \{0\}, \ j \in V \tag{9}$$

$$\max\{0, q_j\} \leq \theta_j \leq \min\{Q, Q+q_j\}, \quad j \in V. \tag{10}$$

Constraints (8) and (9) impose that, if x_{ij} takes value 1, then $\theta_j = \theta_i + q_j$, and hence model the flow conservation independently of the sign of q_j. Constraints (10) simply give lower and upper bounds on the loads.

These constraints generalize the Miller–Tucker–Zemlin subtour elimination constraints for the TSP, see Miller et al. [31], to the case where vertices have demands unrestricted in sign. They can be linearized with the standard "big M" method, obtaining:

$$\theta_j \geq \theta_i + q_j - M(1 - x_{ij}), \quad i \in V, \ j \in V \backslash \{0\} \tag{11}$$

$$\theta_i \geq \theta_j - q_j - M(1 - x_{ij}), \quad i \in V \backslash \{0\}, \ j \in V. \tag{12}$$

Constraints (8) and (9) are linearized to (11) and (12), respectively. In our implementation we set the "big M" equal to $\min\{Q, Q+q_j\}$ in (11), and equal to $\min\{Q, Q-q_j\}$ in (12).

Our *formulation F1* for the BRP is thus given by (1)–(7), (10)–(12). Note that subtour elimination constraints must be kept in the formulation, because (10)–(12) are not enough to ensure connectivity of the solution when some demands are non-positive and some are non-negative.

4.2. Formulation F2

Our second formulation also builds upon the m-TSP, but makes use of an additional variable f_{ij} giving the *flow* over arc (i,j), i.e., the load on the vehicle traveling along arc (i,j), if any, for $(i, j) \in A$. To impose feasibility with respect to the BRP, it is then enough to include in the m-TSP:

$$\sum_{i \in V} f_{ji} - \sum_{i \in V} f_{ij} = q_j, \quad j \in V \backslash \{0\} \tag{13}$$

$$\max\{0, q_i, -q_j\}x_{ij} \leq f_{ij} \leq \min\{Q, Q+q_i, Q-q_j\}x_{ij}, \quad (i,j) \in A. \tag{14}$$

Constraints (13) model the balance of the flows on the arcs entering and leaving a given node. Constraints (14) impose lower and upper bounds on the flows on each arc, and make these bounds as tight as possible by considering whether or not an arc is traveled by a vehicle. Indeed, for what concerns the lower bound, if arc (i,j) is used, then f_{ij} should be at least greater than q_i if $q_i > 0$ (because q_i has just been collected) or greater than $-q_j$ if $q_j < 0$ (because q_j has to be supplied to the next node). Similar considerations apply to the upper bound.

Formulation F2 of the BRP is thus given by (1)–(7), (13) and (14).

4.3. Formulation F3

We use the approach proposed by Hernández-Pérez and Salazar-González [15,16] for the 1-PDTSP in order to develop another formulation of the BRP. We start again from the m-TSP, but ensure that solutions satisfy demands, without exceeding the vehicle capacity, by adding to the model the following family of constraints:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - \max\left\{1, \left\lceil \frac{|\sum_{i \in S} q_i|}{Q} \right\rceil \right\}, \quad S \subseteq V \backslash \{0\}, \ S \neq \varnothing. \tag{15}$$

Constraints (15) are similar to the *generalized subtour elimination constraints* for the CVRP. They state that, for each subset S of vertices, the number of arcs with both tail and head in S should not exceed the cardinality of S minus the minimum number of vehicles required to serve S. An estimation of the minimum number of vehicles is simply obtained by computing the absolute value of the sum of the demands, dividing it by the vehicle capacity and then rounding up the result. This value can be zero when the sum of the q_j is null. In such a case the value 1 is used instead, because at least one vehicle is needed (notice that S does not contain the depot).

Our *formulation F3* is thus given by (1)–(5), (7) and (15). Note that constraints (15) are exponentially many, and hence, as already observed for (6), we need a separation procedure that identifies violated constraints from non-violated ones. Also note that these constraints are valid for formulations F1 and F2, and may be used to improve their convergence to the optimum.

4.4. Formulation F4

We propose a fourth formulation of the BRP that builds upon the two-commodity flow model originally presented by Baldacci et al. [32] for the symmetric CVRP. This formulation first requires to modify the graph by adding a copy of the depot, defined in the following by a new vertex $n+1$. We define $\tilde{V} = V \cup \{n+1\}$ the resulting set of vertices, and $\tilde{A} = A \cup \{(j, n+1) : j \in \tilde{V}\} \cup \{(n+1, j) : j \in \tilde{V}\}$ the resulting set of arcs. Let us also define $\tilde{V}_0 = \tilde{V} \backslash \{0, n+1\}$ and $Q_{tot} = \sum_{j \in \tilde{V}_0} q_j$. We set $q_{n+1} = 0$, $c_{n+1,j} = +\infty$, and $c_{j,n+1} = c_{j0}$ for $j \in \tilde{V}$, knowing that $c_{0,n+1} = 0$, and then we set $c_{j0} = +\infty$ for $j \in \tilde{V}$.

We use three sets of variables. The first two have already been used in formulation F2: x_{ij} takes value 1 if arc (i,j) is used, 0 otherwise, for $(i,j) \in \tilde{A}$, and f_{ij} gives the flow over arc (i,j), i.e., the load of the vehicle passing over arc (i,j), if any, for $(i, j) \in \tilde{A}$. The third variable is another continuous variable g_{ji} that gives the residual space of the vehicle traveling along arc (i,j), if any, for $(i,j) \in \tilde{A}$.

In this formulation, vehicles leave the initial depot, vertex 0, visit customers, and then end their route in the final depot, vertex $n+1$. Variable f_{ij} gives the load along the route performed by the vehicle, starting from the initial depot, vertex 0, following the vertices visited by the route and then ending in the final depot, vertex $n+1$. In the opposite way, g_{ji} gives the free space on the vehicle, starting from $n+1$, retracing back the route and then ending 0. The resulting model is then:

$$\text{(Formulation F4)} \min \sum_{i \in \tilde{V}} \sum_{j \in \tilde{V}} c_{ij} x_{ij} \tag{16}$$

$$\sum_{i \in \tilde{V}} x_{ij} = 1, \quad j \in \tilde{V}_0 \tag{17}$$

$$\sum_{i \in \tilde{V}} x_{ji} = 1, \quad j \in \tilde{V}_0 \tag{18}$$

$$\sum_{j \in \tilde{V}} x_{0j} \leq m \tag{19}$$

$$\sum_{j \in \tilde{V}_0} x_{0j} = \sum_{j \in \tilde{V}_0} x_{j,n+1} \tag{20}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad S \subseteq \tilde{V}_0, \ S \neq \varnothing \tag{21}$$

$$f_{ij} + g_{ji} = Qx_{ij} \quad (i,j) \in \tilde{A} \tag{22}$$

$$\sum_{i \in \tilde{V}} (f_{ji} - g_{ij}) - \sum_{i \in \tilde{V}} (f_{ij} - g_{ji}) = 2q_j, \quad j \in \tilde{V}_0 \tag{23}$$

$$\sum_{j \in \tilde{V}_0} f_{0j} \geq \max\{0, -Q_{tot}\} \tag{24}$$

$$\sum_{j \in \tilde{V}_0} f_{j,n+1} \geq \max\{0, Q_{tot}\} \tag{25}$$

$$\sum_{j \in \tilde{V}_0} g_{j0} \le \min\{mQ, mQ + Q_{tot}\} \tag{26}$$

$$\max\{0, q_i, -q_j\}x_{ij} \le f_{ij} \le \min\{Q, Q+q_i, Q-q_j\}x_{ij} \quad (i,j) \in \tilde{A} \tag{27}$$

$$(Q - \min\{Q, Q+q_i, Q-q_j\})x_{ij} \le g_{ji} \le (Q - \max\{0, q_i, -q_j\})x_{ij} \quad (i,j) \in \tilde{A} \tag{28}$$

$$x_{ij} \in \{0,1\}, \quad i,j \in \tilde{V}. \tag{29}$$

Constraints (17)–(20) and (29) are equivalent to the ones discussed for formulation F1, but adapted to the new sets of vertices and arcs. Constraints (22) state that, if a vehicle travels along arc (i,j), then the sum of the load and the residual space on that arc is equal to the vehicle capacity. Constraints (23) impose that the difference between loads and residual capacities that enter and leave a vertex is double of the demand of that vertex.

The total load leaving the initial depot should be in any case non-negative, and moreover, in case Q_{tot} takes a negative value, it should be not lower than this value. This fact is imposed by constraints (24). Similarly, constraints (25) state that the total load entering the final depot is in any case non-negative, and not lower than the sum of all demands in case this is positive. Inequalities (26) state that the free space on the vehicles leaving the depot must be not greater than the minimum value between the total capacity of the m vehicles and the total capacity plus Q_{tot}. Finally, constraints (27) and (28) enforce lower and upper bounds on the continuous variables.

5. Branch-and-cut algorithm

The models presented in the previous section include an exponential number of constraints, hence we solve them with a *Branch-and-Cut* (B&C) algorithm. We use the B&C framework of CPLEX 12.2, that solves at every node of an enumeration tree the linear relaxation of a MILP model, and then invokes user-developed separation procedures for the possible addition of cuts.

At the root node of the B&C procedure we obtain an initial solution by using a simple greedy heuristic. The greedy method builds routes one at a time, using the principle of the closest neighbor. It starts from the depot and then visits the closest, yet unserved, customer. It then extends the path by moving to the next closest neighbor for which the sum of collected and delivered demands is feasible (see Section 5.2 below for the algorithm used to determine if a path is feasible), if any. When no extension is possible, the route is closed by going back to the depot, and possibly a new route is initialized in an iterative fashion. The model is then initialized with the generalized subtour elimination constraints for sets of one or two vertices, hence $x_{ii} = 0$ for all $i \in V$ and $x_{ij} + x_{ji} \le 2 - \max\{1, \lceil |q_i + q_j|/Q \rceil\}$, for all $(i,j) \in A$. After testing a few branching strategies, we adopted the automatic strong branching procedure by CPLEX that provides the best computational performance on our test bed.

In Section 5.1 we present some valid inequalities that we use to improve the convergence of the algorithm to the optimum, whereas in Section 5.2 we discuss the separation procedures that we implemented. Since inequalities and separation procedures are based only on the x_{ij} variables, they apply directly to formulations 1–3. The adaptation to formulation 4 is trivial, and is outlined in Section 5.2.

5.1. Valid inequalities

We first propose two new simple families of *clique inequalities* with three vertices. Consider a pair of nodes i and j, and let $S(i,j) \subset V \backslash \{0\}$ be the subset of nodes whose demand results in a total greater than Q or smaller than $-Q$. Namely, $S(i,j) = \{h \in V_0, h \ne i, h \ne j : |q_i + q_j + q_h| > Q\}$. Then the following inequalities are valid for the BRP:

$$x_{ij} + \sum_{h \in S(i,j)} x_{jh} \le 1, \quad i,j \in V \backslash \{0\}, \quad h \in S(i,j) \tag{30}$$

$$\sum_{h \in S(i,j)} x_{hi} + x_{ij} \le 1, \quad i,j \in V \backslash \{0\}, \quad h \in S(i,j). \tag{31}$$

We note that, in (30), at most one among the x_{jh} variables may take the value of 1 because of the degree constraint, and that all of them are incompatible with x_{ij} because of the capacity constraint. Thus the sum of all these variables is at most 1 in a feasible solution. A similar consideration applies to (31).

For the next family of inequalities we need some further notation. Let P denote a path consisting of a sequence of vertices starting from the depot. Let also $|P|$ denote the number of vertices in P, and $P(i)$ denote the index of the i-th vertex of path P, for $i = 0, 1, \ldots, |P|-1$, with $P(0) = 0$. Given P, we compute $q_P^{\min} = \min_{i=1}^{|P|-1}\{\sum_{j=1}^{i} q_{P(j)}\}$ and $q_P^{\max} = \max_{i=1}^{|P|-1}\{\sum_{j=1}^{i} q_{P(j)}\}$, that denote, respectively, the minimum and maximum of the cumulative demand along P.

As observed by Hernández-Pérez and Salazar-González [15], a path P is infeasible with respect to the capacity constraint if $q_P^{\max} - q_P^{\min} > Q$. Indeed, if the vehicle leaves the depot with an empty load, then along the path it will reach a maximum load equal to $q_P^{\max}$ that clearly cannot exceed Q. Now notice that $q_P^{\min}$ may be negative even in a feasible solution, but in this case the vehicle has to start from the depot by carrying $-q_P^{\min}$ bikes. Hence, the maximum load becomes $q_P^{\max} - q_P^{\min}$, which also cannot exceed Q.

Let $\mathcal{P}$ be the family of all infeasible paths, the following *infeasible path constraints* are thus valid inequalities for the BRP:

$$\sum_{i=0}^{|P|-2} x_{P(i),P(i+1)} \le |P| - 2, \quad P \in \mathcal{P}. \tag{32}$$

Inequality (32) simply states that, if path P is infeasible, then not all the arcs connecting two consecutive vertices of P may belong to a feasible solution. A way to enforce them is to consider also the arcs connecting non-consecutive vertices of P. Indeed, any arc $(P(i), P(j))$ with $j \ne i+1$ is incompatible with arcs $(P(i), P(i+1))$ and $(P(j-1), P(j))$, because of, respectively, out-degree and in-degree constraints. Hence we can add the corresponding variable, $x_{P(i),P(j)}$ to the left-hand side of (32), without affecting the right-hand side value. This process can be repeated for all arcs connecting two non-consecutive vertices in the path, with the exception of those arcs leaving the depot, because up to m of them may belong to a feasible solution. Hence, we obtain the following:

$$x_{0,P(1)} + \sum_{i=1}^{|P|-2} \sum_{j=i+1}^{|P|-1} x_{P(i),P(j)} \le |P| - 2, \quad P \in \mathcal{P}. \tag{33}$$

Inequalities (33) generalize, in the case of multiple vehicles, the *tournament constraints*, discussed by Ascheuer et al. [33] for the Traveling Salesman Problem. Note that these constraints can be used in directed formulations (where a variable is associated to each directed arc), but cannot be applied to undirected ones (where variables are associated to undirected edges).

5.2. Separation procedures

The aim of this section is to present the procedures that we use to determine whether or not the valid inequalities we proposed in Section 5.1 are violated by a given solution $\bar{x}$ (possibly fractional).

Separations S1 and S2: The inequalities of type (30) and (31) are separated exactly, by enumerating all possible pairs of vertices i and j, computing the set S_{ij}, and then evaluating if the sum of the $\bar{x}$ involved in the inequality exceeds one. In the following we will refer the separation of (30) and (31) as *separation S1* and *separation S2*, respectively.

Separation S3: To separate constraints (6) and (15), we first build a supporting graph $\overline{G} = (\overline{V}, \overline{A})$, where $\overline{V} = V$, $\overline{A} = \{(i,j) \in A : \bar{x}_{ij} > 0\}$, and a capacity $\bar{x}_{ij}$ is assigned to every arc $(i,j) \in \overline{A}$. Constraints (6) may be separated exactly as usually done for the TSP, by computing $O(n)$ max flows on $\overline{G}$, using the depot as a source and any vertex i, in turn, as a sink. If the max flow value obtained is lower than one, then i is disconnected from the depot, and hence the cut that corresponds to the set S induced by the min cut may be added to the model. For the BRP, we obtain a simple algorithmic improvement by checking, for any set S induced by a min cut, if the minimum number of vehicles required to serve S is lower than the max flow value obtained, and, in such a case, we add the corresponding (stronger) constraint (15). In this way the exact separation of (6) is used as a heuristic for separating (15). We refer to this procedure as *separation S3*.

Separation S4: To separate constraint (15), we extend the procedure developed in Hernández-Pérez and Salazar-González [15] for the 1-PDTSP to the directed case. First of all we associate a demand $q_0 = Q_{tot}$ to the depot. Then we build a new supporting graph $\overline{G}'$, obtained by adding two dummy nodes, $n+1$ and $n+2$, to $\overline{G}$. We connect $n+1$ to any $i \in V$ having $q_i > 0$ with an arc of capacity q_i/Q, and then connect any $i \in V$ having $q_i < 0$ to $n+2$ with an arc of capacity $-q_i/Q$ (in this way, the capacities associated with the arcs are always non-negative). We then compute the max flow on $\overline{G}'$, using $n+1$ as a source and $n+2$ as a sink. The constraint (15) corresponding to the set S induced by the min cut is then checked, and, if violated, it is added to the model. We refer to Hernández-Pérez and Salazar-González [15] for further details.

Conforming to the fact that we are solving a problem on an asymmetric graph, we perform a second separation for (15). We build a second modified supporting graph $\overline{G}''$, adding again two new dummy vertices, $n+1$ and $n+2$, to $\overline{G}$. In this case however we include in $\overline{G}''$ new arcs $(n+1, i)$ of capacity $-q_i/Q$ for all $i \in V$ having $q_i < 0$, and new arcs $(i, n+2)$ of capacity q_i/Q for all $i \in V$ having $q_i > 0$. Once more, we use the max flow procedure on the resulting supporting graph, from $n+1$ to $n+2$, and check the constraints (15) for the sets S induced by the min cuts. In the following, these procedures are referred to as *separation S4*.

Separation S5: The tournament constraints (33) are separated exactly, by generating all possible paths starting from the depot and using a depth-first strategy. We initialize vector P with $P(0) = 0$, then select the first outgoing arc having a positive value of $\bar{x}$, say, $(0, i)$, and extend the path to include the head of the selected arc by setting $P(1) = i$. The process is then reiterated. Every time we add a node to the path, we check if it is infeasible. If so, then we add the cut and backtrack to the previous node, otherwise we continue extending the path. The path extension continues as long as the sum of the involved $\bar{x}_{ij}$ is large enough to possibly lead to a violated cut, i.e., as long as it is strictly greater than $|P|-2$. As soon as the sum becomes smaller, we backtrack to the previous node. Anytime we backtrack, we continue the depth-first search by selecting the next arc with positive value of $\bar{x}$ and then extend the path consequently.

Suppose a violated cut of type (33) has been found, then we know that the set $S = \{P(1), P(2), \ldots, P(|P|)\}$ is currently visited by a single vehicle (the one performing the path P). Now we estimate the minimum number of vehicles required to serve S using $\lceil |\sum_{i \in S} q_i|/Q \rceil$, and, if this is greater than one, we add the violated generalized subtour elimination constraint (15) for this set S. Constraints (15) are usually stronger than (33), but do not dominate them (because the latter involves variable $x_{0,P(1)}$, not contained in the former), and hence they are both profitable for the model solution. The current procedure is called *separation S5*.

The separation procedures (S1–S5) are invoked at every node of the enumeration tree in the order in which we described them. With regards to formulation F4, there is only a slight modification

Table 3
Implementation of formulations F1–F4.

Formulation	Variables	Core formulation	Additional separations
F1	x_{ij}, θ_j	(1)–(7), (10)–(12), S3	S1, S2, S4, S5
F2	x_{ij}, f_{ij}	(1)–(7), (13) and (14), S3	S1, S2, S4, S5
F3	x_{ij}	(1)–(5), (7) and (15), S3, S4	S1, S2, S5
F4	x_{ij}, f_{ij}, g_{ij}	(16)–(29), S3	S1, S2, S4, S5

Table 2
Benchmark instances (notation explained in Table 1).

City	Country	$\|V\|$	min$\{q_i\}$	avg$\{q_i\}$	max$\{q_i\}$	dev$\{q_i\}$	min$\{c_{ij}\}$	avg$\{c_{ij}\}$	max$\{c_{ij}\}$	dev$\{c_{ij}\}$
Bari	Italy	13	−5	−1.54	5	2.70	400	2283.97	5400	1067.62
Reggio Emilia	Italy	14	−10	−2.00	3	4.17	300	2095.05	5500	1110.44
Bergamo	Italy	15	−12	−1.00	10	5.39	100	1532.86	3200	631.94
Parma	Italy	15	−6	−1.07	4	2.76	200	3121.43	8800	1857.86
Treviso	Italy	18	−4	−0.83	3	2.23	340	3510.99	8398	2133.37
La Spezia	Italy	20	−5	0.05	6	2.93	193	2521.61	6128	1279.82
Buenos Aires	Argentina	21	−20	−0.43	20	17.15	689	4676.75	12 780	2246.80
Ottawa	Canada	21	−5	−0.05	5	2.58	180	2219.15	5030	1012.63
San Antonio	US	23	−4	1.74	8	3.43	98	1950.98	4808	944.61
Brescia	Italy	27	−11	−1.41	4	3.69	200	2571.65	6600	1174.49
Roma	Italy	28	−17	−2.36	18	9.45	200	5989.55	27 400	7361.37
Madison	US	28	−6	0.29	8	2.93	53	3085.99	9922	1780.75
Guadalajara	Mexico	41	−11	−1.07	1	1.94	60	3278.30	14 728	2440.40
Dublin	Ireland	45	−11	−1.42	6	3.73	203	2190.50	4734	933.16
Denver	US	51	−8	−0.69	7	3.28	211	3873.94	12 000	2643.54
Rio de Janeiro	Brazil	55	−7	−1.47	7	3.82	420	5328.35	16 591	2587.97
Boston	US	59	−8	−0.27	16	5.28	243	3911.82	19 239	2378.22
Torino	Italy	75	−7	−0.49	9	3.73	23	2527.84	7200	1157.14
Toronto	US	80	−11	0.15	12	5.17	150	2339.03	6283	1103.70
Miami	US	82	−8	−2.24	9	3.99	68	4000.00	13 771	3336.49
Ciudad de Mexico	Mexico	90	−17	−0.97	17	8.48	15	2551.94	7264	1409.34
Minneapolis	US	116	−9	−0.79	5	2.87	6	6045.24	19 468	3558.17

Table 4
Computational results for formulations F1–F4 on all problem instances.

Inst.	City ($\|V\|$, Q)	*UB*	F1		F2		F3		F4	
			Gap_1	Time	Gap_2	Time	Gap_3	Time	Gap_4	Time
1	Bari (13, 30)	14 600	0.00	0.06	0.00	0.11	0.00	0.02	0.00	0.27
2	Bari (13, 20)	15 700	0.00	0.06	0.00	0.10	0.00	0.02	0.00	0.11
3	Bari (13, 10)	20 600	0.00	0.16	0.00	0.39	0.00	0.03	0.00	0.32
4	Reggio Emilia (14, 30)	16 900	0.00	0.03	0.00	0.03	0.00	0.02	0.00	0.06
5	Reggio Emilia (14, 20)	23 200	0.00	0.09	0.00	0.52	0.00	0.03	0.00	0.33
6	Reggio Emilia (14, 10)	32 500	0.00	5.59	0.00	0.67	0.00	0.05	0.00	0.22
7	Bergamo (15, 30)	12 600	0.00	0.05	0.00	0.05	0.00	0.03	0.00	0.10
8	Bergamo (15, 20)	12 700	0.00	0.06	0.00	0.26	0.00	0.00	0.00	0.21
9	Bergamo (15, 12)	13 500	0.00	0.27	0.00	0.74	0.00	0.11	0.00	0.95
10	Parma (15, 30)	29 000	0.00	0.05	0.00	0.03	0.00	0.02	0.00	0.06
11	Parma (15, 20)	29 000	0.00	0.05	0.00	0.02	0.00	0.02	0.00	0.06
12	Parma (15, 10)	32 500	0.00	0.22	0.00	0.26	0.00	0.05	0.00	0.45
13	Treviso (18, 30)	29 259	0.00	0.12	0.00	0.14	0.00	0.05	0.00	0.62
14	Treviso (18, 20)	29 259	0.00	0.12	0.00	0.20	0.00	0.03	0.00	0.45
15	Treviso (18, 10)	31 443	0.00	0.27	0.00	0.75	0.00	0.09	0.00	0.79
16	La Spezia (20, 30)	20746	0.00	0.09	0.00	0.16	0.00	0.03	0.00	0.31
17	La Spezia (20, 20)	20 746	0.00	0.09	0.00	0.10	0.00	0.03	0.00	0.24
18	La Spezia (20, 10)	22 811	0.00	0.16	0.00	1.45	0.00	0.09	0.00	1.34
19	Buenos Aires (21, 30)	76 999	0.00	1.36	0.00	3.54	0.00	0.37	0.00	12.40
20	Buenos Aires (21, 20)	91 619	0.00	23.26	0.00	3.58	0.00	16.80	0.00	7.35
21	Ottawa (21, 30)	16 202	0.00	0.06	0.00	0.05	0.00	0.02	0.00	0.09
22	Ottawa (21, 20)	16 202	0.00	0.06	0.00	0.06	0.00	0.02	0.00	0.09
23	Ottawa (21, 10)	17 576	0.00	0.30	0.00	1.90	0.00	0.11	0.00	1.26
24	San Antonio (23, 30)	22 982	0.00	0.19	0.00	2.74	0.00	0.08	0.00	4.38
25	San Antonio (23, 20)	24 007	0.00	3.63	0.00	1.36	0.00	0.09	0.00	7.52
26	San Antonio (23, 10)	40 149	5.36	3600.00	0.00	7.07	0.00	1.06	0.00	17.86
27	Brescia (27, 30)	30 300	0.00	0.70	0.00	0.77	0.00	0.06	0.00	2.22
28	Brescia (27, 20)	31 100	0.00	6.07	0.00	6.38	0.00	0.20	0.00	9.11
29	Brescia (27, 11)	35 200	0.00	24.46	0.00	10.83	0.00	1.37	0.00	9.31
30	Roma (28, 30)	61 900	0.00	4.27	0.00	5.48	0.00	0.84	0.00	8.46
31	Roma (28, 20)	66 600	0.00	22.04	0.00	7.61	0.00	1.72	0.00	33.38
32	Roma (28, 18)	68 300	0.00	16.15	0.00	3.26	0.00	0.58	0.00	4.99
33	Madison (28, 30)	29 246	0.00	0.09	0.00	0.10	0.00	0.02	0.00	0.24
34	Madison (28, 20)	29 839	0.00	0.31	0.00	0.44	0.00	0.05	0.00	0.84
35	Madison (28, 10)	33 848	0.00	6.02	0.00	16.56	0.00	0.53	0.00	6.36
36	Guadalajara (41, 30)	57 476	0.00	1.16	0.00	2.97	0.00	0.22	0.00	6.22
37	Guadalajara (41, 20)	59 493	0.00	2.29	0.00	8.18	0.00	0.34	0.00	7.21
38	Guadalajara (41, 11)	64 981	1.18	3600.00	0.00	39.15	0.00	1.79	0.00	17.99
39	Dublin (45, 30)	33 548	0.00	435.69	0.00	180.81	0.00	6.05	0.00	280.05
40	Dublin (45, 20)	39 786	2.43	3600.00	1.14	3600.00	0.00	76.72	0.00	3140.89
41	Dublin (45, 11)	54 392	7.70	3600.00	1.17	3600.00	0.00	610.80	0.00	3513.55
42	Denver (51, 30)	51 583	0.00	66.57	0.00	11.01	0.00	0.67	0.00	15.88
43	Denver (51, 20)	53 465	0.00	410.05	0.00	372.78	0.00	25.33	0.00	1096.95
44	Denver (51, 10)	67 459	1.47	3600.00	0.00	1535.62	0.00	231.52	0.00	3443.68
45	Rio de Janeiro (55, 30)	122 547	4.16	3600.00	1.57	3600.00	0.00	65.57	1.97	3600.00
46	Rio de Janeiro (55, 20)	156 140	8.57	3600.00	2.29	3600.00	0.44	3600.00	1.82	3600.00
47	Rio de Janeiro (55, 10)	259 049	28.58	3600.00	2.07	3600.00	2.23	3600.00	2.23	3600.00
48	Boston (59, 30)	65 669	0.00	255.25	0.00	1136.46	0.00	28.14	0.00	975.97
49	Boston (59, 20)	71 879	0.00	1899.07	1.70	3600.00	0.00	473.84	1.63	3600.00
50	Boston (59, 16)	75 065	0.99	3600.00	2.59	3600.00	0.37	3600.00	2.94	3600.00
51	Torino (75, 30)	47 634	0.00	464.46	0.00	919.59	0.00	13.79	0.00	1703.46
52	Torino (75, 20)	50 204	2.38	3600.00	2.70	3600.00	0.00	859.69	3.50	3600.00
53	Torino (75, 10)	64 797	11.30	3600.00	9.66	3600.00	9.25	3600.00	9.23	3600.00
54	Toronto (80, 30)	41 549	3.54	3600.00	4.47	3600.00	1.82	3600.00	4.42	3600.00
55	Toronto (80, 20)	47 898	11.81	3600.00	11.02	3600.00	11.05	3600.00	11.39	3600.00
56	Toronto (80, 12)	60 763	13.15	3600.00	10.74	3600.00	11.84	3600.00	10.88	3600.00
57	Miami (82, 30)	156 104	23.68	3600.00	8.43	3600.00	2.48	3600.00	9.01	3600.00
58	Miami (82, 20)	229 237	38.43	3600.00	11.02	3600.00	8.66	3600.00	10.99	3600.00
59	Miami (82, 10)	415 762	38.14	3600.00	6.07	3600.00	6.49	3600.00	6.12	3600.00
60	Ciudad de Mexico (90, 30)	88 227	28.67	3600.00	23.28	3600.00	23.05	3600.00	23.74	3600.00
61	Ciudad de Mexico (90, 20)	116 418	29.95	3600.00	23.59	3600.00	23.61	3600.00	23.63	3600.00
62	Ciudad de Mexico (90, 17)	109 573	20.00	3600.00	9.00	3600.00	9.56	3600.00	10.73	3600.00
63	Minneapolis (116, 30)	137 843	4.58	3600.00	5.82	3600.00	1.23	3600.00	5.96	3600.00
64	Minneapolis (116, 20)	186 449	23.45	3600.00	18.61	3600.00	15.40	3600.00	18.41	3600.00
65	Minneapolis (116, 10)	298 886	37.08	3600.00	17.65	3600.00	17.96	3600.00	18.93	3600.00
Average gaps and times			5.33	1330.02	2.69	1228.99	2.24	923.37	2.73	1272.84
Opt			**42**		**44**		**49**		**46**	
#Nodes			2134.32		1847.74		1240.91		1440.65	
#Cuts			2233.02		630.14		1908.28		578.09	

to implement. Recall that this formulation works on a modified graph $\tilde{V}$, where $n+1$ is a copy of the original depot 0. When creating the supporting graph $\overline{G}$, we merge again 0 and $n+1$ into a unique vertex 0 (this is done by simply setting $\overline{x}_{j,0} = \overline{x}_{j,n+1}$ for all $j \in V$), and then use the separation process just described for the previous formulations. The resulting cut, if any, is then mapped back on $\tilde{V}$. On the basis of computational evidence resulting from detailed experimentation, the separation process for all formulations is terminated as soon as a violated cut is found.

6. Benchmark instances

In our attempt to solve real-world instances, we collected real data from the web sites of twenty-two bike sharing systems characterized by diverse size. The cities included in our studies are: Bari, Brescia, Bergamo, La Spezia, Parma, Rome, Torino, Treviso, and Reggio Emilia, in Italy; Dublin in Ireland; Boston, Denver, Madison, Miami, Minneapolis, and San Antonio in the USA; Ottawa and Toronto in Canada; Ciudad de Mexico and Guadalajara in Mexico; Buenos Aires in Argentina and Rio de Janeiro in Brazil.

We have been in touch with the bike sharing operators of these cities, in particular with those people responsible for the bike repositioning, to obtain information regarding the depot location, the desirable level of occupation, the characteristics of the available fleet of vehicles and the type of repositioning performed. Not all of them furnished us with all the required information. In general, we can state that some of them perform the repositioning during the night, some others also during the day. From the web sites we collected the coordinates of the stations and of the depot. If no information for the depot was available, we assumed that the depot was located at the same place as one of the stations. We used a Geographical Information System to compute the shortest traveling distances c_{ij} (in meters) between each pair of points, considering the two possible directions. We then took a snapshot of the nightly level of occupation at each station, and fixed the demand of the station to be the difference between the level of occupation encountered and the desired level of occupation of the station. Indeed, according to the information provided by the majority of operators, the most common rule adopted to rebalance the stations is to fill half of the slots in each station, thus we set the desired level of occupation of a station to half its number of slots. Also note that, for those cities where the depot was allocated to a bike station, we set the demand of that station to 0.

According to the data we collected, the most common vehicle capacities encountered in practice are 30, 20 and 10 bikes for each vehicle. We applied these three values to the data obtained for each city. For some instances the minimum vehicle capacity used is set to $\max_{i \in V}\{|q_i|\}$, when the number of the latter exceeds 10. For Buenos Aires, whose maximum demand was 20, we tested only 20 and 30. We generated 65 instances in total that are summarized in Table 2. The values $\min\{q_i\}$, $\text{avg}\{q_i\}$, $\max\{q_i\}$, and $\text{dev}\{q_i\}$ give, respectively, the minimum, average, and maximum demand, and the standard deviation. Similar information is provided for the distances c_{ij}. The number of vertices varies from 13 to 116 stations. All test instances have variability in the number

Table 5
Root node – comparison of various separation procedures on the core formulation F3.

City ($\|V\|$, Q)	Gap_{core}	Gap^+(S1)	Gap^+(S2)	Gap^+(S5)
Dublin (45 ,30)	1.45	1.45	1.45	1.45
Dublin (45 , 20)	4.56	4.31	3.68	4.56
Dublin (45, 11)	5.52	5.61	4.69	4.86
Denver (51, 30)	0.52	0.52	0.52	0.52
Denver (51, 20)	2.58	2.58	2.58	2.56
Denver (51, 10)	3.34	3.05	4.29	3.32
Rio de Janeiro (55, 30)	2.05	2.05	2.05	2.05
Rio de Janeiro (55, 20)	2.51	2.51	2.51	2.51
Rio de Janeiro (55, 10)	4.12	3.99	3.35	4.12
Boston (59, 30)	1.87	1.76	1.76	1.87
Boston (59, 20)	3.31	3.45	3.11	3.28
Boston (59, 16)	4.63	4.25	4.13	4.63
Torino (75, 30)	1.35	1.35	1.35	1.35
Torino (75, 20)	3.38	3.81	3.30	3.38
Torino (75, 10)	11.73	11.44	11.50	11.73
Toronto (80, 30)	5.55	5.55	5.55	5.55
Toronto (80, 20)	15.19	13.48	14.10	15.19
Toronto (80, 12)	14.40	14.87	14.97	14.40
Miami (82, 30)	3.28	3.28	3.28	3.28
Miami (82, 20)	8.91	9.17	9.29	8.91
Miami (82, 10)	6.71	7.03	6.86	6.71
Ciudad de Mexico (90, 30)	24.62	25.23	25.18	24.62
Ciudad de Mexico (90, 20)	25.43	24.96	24.94	25.43
Ciudad de Mexico (90, 17)	11.81	11.31	11.85	11.80
Minneapolis (116, 30)	2.97	2.97	2.97	2.97
Minneapolis (116, 20)	15.97	15.97	15.97	15.97
Minneapolis (116, 10)	19.44	19.14	18.83	19.44
Average gaps (%)	7.67	7.60	7.56	7.65

Table 6
Root node – comparison of various separation procedures on the full formulation F3.

City ($\|V\|$, Q)	Gap_{full}	Gap^- (S1)	Gap^- (S2)	Gap^- (S3)	Gap^- (S4)	Gap^- (S5)
Dublin (45 ,30)	1.45	1.45	1.45	3.75	3.47	1.45
Dublin (45 , 20)	3.68	3.68	4.31	6.38	12.08	3.68
Dublin (45, 11)	5.16	4.43	5.61	5.26	13.95	5.16
Denver (51, 30)	0.52	0.52	0.52	1.31	1.64	0.52
Denver (51, 20)	2.56	2.56	2.56	2.60	3.33	2.58
Denver (51, 10)	3.49	4.29	2.98	4.82	6.65	3.49
Rio de Janeiro (55, 30)	2.05	2.05	2.05	2.05	6.45	2.05
Rio de Janeiro (55, 20)	2.51	2.51	2.51	2.51	10.56	2.51
Rio de Janeiro (55, 10)	3.30	3.35	3.41	3.30	21.50	3.30
Boston (59, 30)	1.76	1.76	1.76	2.75	1.68	1.76
Boston (59, 20)	3.45	3.11	3.45	4.08	3.88	3.45
Boston (59, 16)	3.87	4.13	4.25	4.48	4.33	3.87
Torino (75, 30)	1.35	1.35	1.35	2.50	1.61	1.35
Torino (75, 20)	3.81	3.30	3.81	4.68	4.41	3.81
Torino (75, 10)	11.56	11.50	11.44	12.84	15.15	11.56
Toronto (80, 30)	5.55	5.55	5.55	7.25	6.08	5.55
Toronto (80, 20)	13.30	14.10	13.48	14.62	14.90	13.30
Toronto (80, 12)	14.99	14.97	14.87	15.36	18.99	14.99
Miami (82, 30)	3.28	3.28	3.28	3.36	20.32	3.28
Miami (82, 20)	9.14	9.25	9.14	9.17	28.36	9.17
Miami (82, 10)	7.00	6.86	7.03	7.01	35.36	7.00
C. de Mexico (90, 30)	25.46	25.18	25.23	25.51	32.58	25.46
C. de Mexico (90, 20)	25.04	24.94	24.96	25.04	34.97	25.04
C. de Mexico (90, 17)	11.48	11.48	11.31	11.50	24.47	11.55
Minneapolis (116, 30)	2.97	2.97	2.97	3.59	6.73	2.97
Minneapolis (116, 20)	15.97	15.97	15.97	16.50	23.42	15.97
Minneapolis (116, 10)	18.52	18.83	19.14	18.72	36.39	18.83
Average gaps (%)	7.53	7.53	7.57	8.18	14.56	7.54

of bikes leaving the depot (since avg $\{q_i\}$ is not 0). All distances are given in meters. All instances are available online at http://www.or.unimore.it/resources.htm.

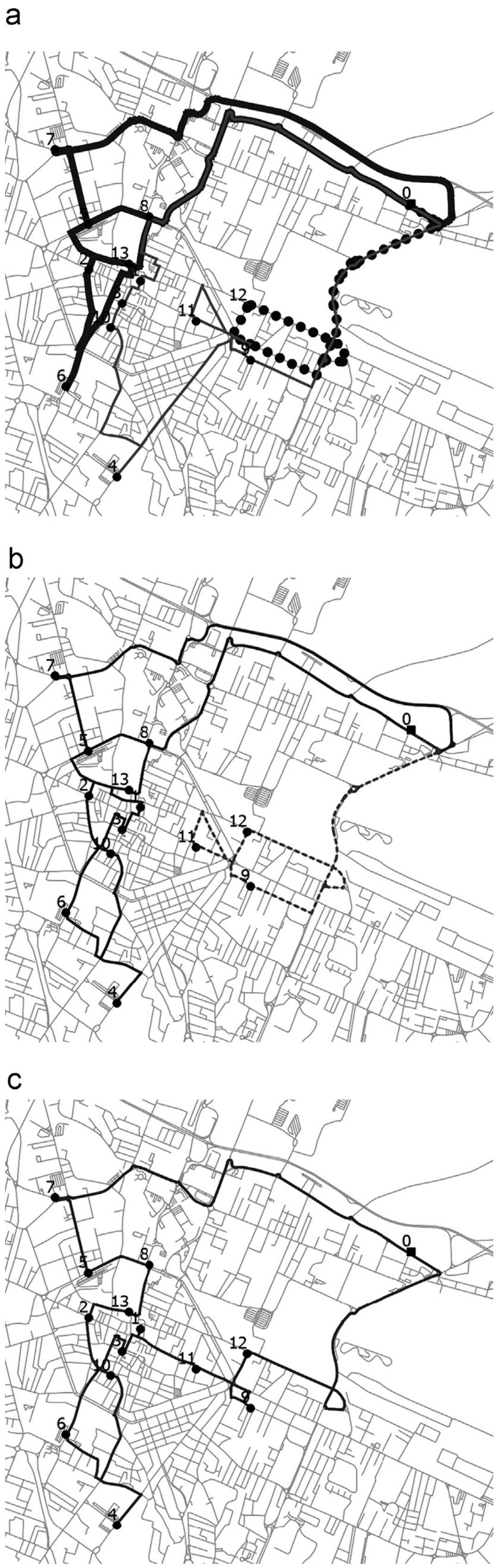

Fig. 4. Optimal solutions for Reggio Emilia with (a) $Q=10$, (b) $Q=20$, and (c) $Q=30$.

7. Computational results

All algorithms described in Sections 4 and 5 were coded in C/C++ and run on an Intel Core i3-2100 CPU, 3.10 GHz, 4.00 GB. We used CPLEX 12.2 as the MILP solver imposing the selection of a single processor. Formulations F1–F4 were computationally tested on the problem instances described in Section 6. For clarity purposes, details of the computational implementation of the formulations we tested are given in Table 3, where we also include the additional separation procedures described in Section 5.2.

In this section, we present the computational results of the B&C algorithm using formulations F1–F4. The B&C algorithm was terminated when either the optimal solution was found or a predetermined time-limit elapsed. The time-limit imposed was 3600 CPU seconds. Table 4 reports the computational results obtained for the complete set of 65 problem instances. In this table, the following information is provided for each problem instance:

- *City* $(|V|, Q)$: City name associated with a given problem instance which is uniquely identified by the number of vertices $|V|$ of the corresponding digraph and the vehicle capacity Q used to transport the bikes at the specified city.
- *UB*: The value of the best feasible solution found by the B&C algorithm using formulations F1–F4.
- Gap_1–Gap_4: Percentage deviation from *UB* of LB_1–LB_4, best lower bounds obtained by the B&C algorithm using core formulations F1–F4 including the set of additional separations for each formulation, respectively, as shown in Table 3. $Gap_i = 100/(1-LB_i/UB)$, for $i=1,\dots,4$.
- *Time*: Computational time for running the B&C algorithm, in CPU seconds.

To compare the average performance of formulations F1–F4 over the complete set of problem instances, the following additional information is given in Table 4:

- *Opt*: The total number of problem instances solved to optimality by the B&C algorithm within 3600 s.
- *#Nodes*: Average number of nodes (over all instances) explored by the B&C algorithm.
- *#Cuts*: Average number of cuts generated by the B&C algorithm.

As shown in Table 4, the first 25 problem instances and a large number of other instances are easily solved by all formulations within a few seconds. Formulation F1 gives the worst results in terms of average gap, computational time, and number of generated cuts, whereas, formulations F2 and F4 demonstrate similar performance for all performance indicators. On the other hand, F3 performs best over all formulations with average $Gap_3 = 2.24\%$ and it optimally solves all problem instances including up to 50 vertices. Indeed, formulation F3 finds the optimal solution for 49 out of 65 instances within an average time of 15 min generating an average of approximately 1200 B&C nodes. However, it is worth noting that, for some of the larger instances, Gap_3 is larger than Gap_2 (e.g., Ciudad de Mexico (90, 17)) or Gap_4 (e.g., Miami (82, 10)). Further insight into the best performing formulation F3 was gained by computationally evaluating the impact of each separation procedure on the resulting percentage gaps computed at the root node of the B&C tree for the most difficult problem instances.

In Table 5 we test the core formulation F3, see Table 3, and the ones obtained by including just one of the separation procedures at a time. We denote with Gap_{core} the percentage deviation from *UB* of the lower bound obtained with the core version, while we use $Gap^{+}(Sx)$ to denote the results of the core version *plus* separation *Sx*.

In Table 6 we present the results of the full version of formulation F3 (i.e., core formulation plus all separation procedures) and those obtained by removing from it one separation procedure at a time. We denote with Gap_{full} the percentage deviation from *UB* of the lower bound obtained with the full version, while we use $Gap^{-}(Sx)$ to denote the results of the full version *minus* separation *Sx*.

The results reported in Table 5 demonstrate that the average gap of the core formulation F3 is close to 7% but slightly greater than 25% in the worst case – Ciudad De Mexico (90, 20). The average gap Gap_{core} is only slightly improved by separately adding either separation S1, S2 or S5; however, it is worth noting that separations S1, S2 and S5 are computed very fast. The importance of separations S3 and, more clearly, S4 is evident from the results presented in Table 6, namely, removing S3 and S4 from the full formulation F3 leads to an increase of the percentage gap from an average value of $Gap_{full} = 7.53\%$ to $Gap^{-}(S3) = 8.18\%$ and $Gap^{-}(S4) = 14.56\%$, respectively. Notably, this gap increases from 18.52% to 36.39% for separation S4 in the case of Minneapolis (116, 10).

Based on the overall computational results presented in this section, we note that the computational difficulty of the instances grows not only with the number of vertices, as it is expected, but also when the capacity of the vehicles decreases. This behavior was also noted for the 1-PDTSP by Hernández-Pérez and Salazar-González [15] and is due to the fact that small capacity vehicles are forced to perform long and complicated routes for pickups and deliveries. An illustrative example is given in Fig. 4, which depicts the three optimal solutions obtained for the problem instance arising in Reggio Emilia for the following cases:

(i) $Q=10$, three vehicle routes are needed. The first one visits stations 2, 6, 13, 5, 8 and 7, the second vehicle visits stations 1, 3, 10, 4, 9, and 11 but the third one visits only station 12. The total distance traveled is 32.5 km.
(ii) $Q=20$, two vehicle routes are needed. The first one visits stations 13, 2, 10, 4, 6, 3, 1, 5, 8 and 7 while stations 9, 11, and 12 are visited by the second vehicle. Total distance traveled is 23.2 km.
(iii) $Q=30$, a single vehicle route visits all stations (traveling for 16.9 km) in the following sequence: 7, 5, 8, 13, 2, 10, 4, 6, 3, 1, 11, 9, and 12.

7.1. Randomly generated instances

It is well known that TSP formulations perform differently on clustered and randomized data set. To this end, we created a set of random instances, in addition to the real-world ones, to evaluate the quality of our algorithm. We followed a classical random generation method adopted in the CVRP literature, see e.g. Toth and Vigo [34]. We randomly generated instances including 40, 45, 50, 55 and 60 vertices, because problems of this size are challenging to solve for real-world cases. The coordinates of the vertices are randomly generated between 0 and 100, while the depot is set in (50, 50). Distances between vertices are computed as the Euclidean ones, but perturbed by a factor δ, randomly chosen between -20% and 20% to induce asymmetry on the costs. In more detail, let b_{ij} be the Euclidean distance between the two vertices i and j, for $i, j \in V, j > i$. The cost matrix is computed by setting $c_{ij} = b_{ij}(1+\delta)$, for $i, j \in V$, then the resulting c_{ij} values are multiplied by 100 and rounded up to the next integer value. A check on triangularity of distances is then performed to adjust non-triangular distances. We considered 10, 20 and 30 as plausible values for *Q*. The demand of each vertex is randomly generated between -10 and 10 but other than zero, while the demand of the depot is set to zero.

In Table 7 we report the instance name, the number of vertices, the vehicle capacity, the best upper bound found and, for each formulation, the percentage gap between the best upper bound and the best lower bound, and the solution time. The average gaps and computational times are also reported. The number of instances solved to optimality is indicated by *Opt*, while *#Nodes* and *#Cuts* illustrate the average number of visited nodes and added cuts for each formulation. The time limit was set to 3600 s.

The results confirm the computational experience derived from solving the real-world instances. When increasing the number of vertices and the vehicle capacity the instances get harder to solve. The formulation having better results is, once again, F3. As reported in Table 7, 12 instances out of 15 are solved to optimality by F3, and the percentage gap between the best upper bound and the best lower bound is, on average, lower than 0.20%. The average solution time is less than 15 min. Overall, tests on randomly generated instances confirm the quality of the proposed algorithms and formulations.

Table 7
Tests on randomly generated instances.

Name	$\|V\|$	Q	UB	F1		F2		F3		F4	
				Gap_1	Time	Gap_2	Time	Gap_3	Time	Gap_4	Time
R40-30	40	30	59 429	1.45	3600.00	0.00	451.32	0.00	13.23	0.00	241.58
R40-20	40	20	67 789	1.92	3600.00	0.00	1171.56	0.00	115.25	0.00	845.35
R40-10	40	10	102 941	3.85	3600.00	0.00	890.31	0.00	217.83	0.00	842.11
R45-30	45	30	52 980	0.00	70.23	0.00	42.42	0.00	2.81	0.00	115.88
R45-20	45	20	58 204	0.00	825.68	0.00	1860.36	0.00	56.08	0.63	3600.00
R45-10	45	10	75 223	2.42	3600.00	2.84	3600.00	0.00	591.15	3.18	3600.00
R50-30	50	30	60 960	0.00	14.59	0.00	52.95	0.00	7.83	0.00	210.09
R50-20	50	20	67 212	0.00	2348.79	1.19	3600.00	0.00	41.82	0.95	3600.00
R50-10	50	10	94 465	4.55	3600.00	1.23	3600.00	0.00	248.73	0.28	3600.00
R55-30	55	30	67 491	0.00	872.29	0.00	403.45	0.00	11.83	0.00	1155.85
R55-20	55	20	74 429	0.00	2744.08	0.00	1143.11	0.00	33.71	0.00	1333.42
R55-10	55	10	103 117	12.37	3600.00	1.86	3600.00	1.07	3600.00	2.17	3600.00
R60-30	60	30	76 574	6.15	3600.00	3.08	3600.00	0.00	1025.23	3.52	3600.00
R60-20	60	20	89 762	9.96	3600.00	4.39	3600.00	0.88	3600.00	3.27	3600.00
R60-10	60	10	133 824	12.95	3600.00	1.90	3600.00	0.78	3600.00	1.78	3600.00
Average gaps and times				3.71	2618.38	1.10	2081.03	0.18	877.70	1.05	2236.28
Opt				**6**		**8**		**12**		**7**	
#Nodes				7741.20		6479.47		4664.30		4958.07	
#Cuts				2822.73		903.27		1145.80		758.67	

8. Conclusions and future research directions

In this paper we considered the Bike Rebalancing Problem, which calls for the repositioning of public bikes of a bike sharing program at minimum cost, using a fleet of capacitated vehicles. We modeled the problem proposing four different formulations, which take into account different types of variables and constraints. All formulations involve an exponential number of constraints, so we solved them by using a branch-and-cut algorithm.

By collecting data from several web sites, we produced an interesting test bed containing 65 instances. We made the test bed publicly available, and used it to computationally evaluate the branch-and-cut algorithm. Computational evidence suggests that the best computational results are produced by formulation F3. This formulation efficiently solves instances with up to 50 vertices, but may fail in solving some larger ones in 1 h of computational time on a standard PC. An interesting future research direction is to try to solve these instances by means of heuristic and metaheuristic algorithms.

Acknowledgments

We thank the Mobility Concilorship of Reggio Emilia, and Alessandro Meggiato, Mobility Manager of the Municipality of Reggio Emilia, for the data and information supplied. We also thank Scott Mullen (General Manager of Hubway, Boston bike sharing system), Matt Verlee (Customer Relations & Low Income Outreach Manager of Denver B-Cycle), Marco Verità (of Brescia Mobilità, Brescia bike sharing system), Ben Kunde (Tech Manager, Madison B-Cycle), Caleb Cohate (San Antonio B-Cycle), Sonia Fakiel (Mejor en Bici, Buenos Aires bike sharing system) and ATB Mobilità (La BiGi, Bergamo bike sharing system) for the information they furnished us with.

We thank Ministero dell'Istruzione, dell'Università e della Ricerca, Italy, for financial support. Manuel Iori is partially supported by CAPES/Brazil under Grant PVE no. A007/2013.

We are grateful to the associate editor and to two anonymous referees for their valuable comments.

References

[1] DeMaio P. Bike sharing: history, impacts, model of provision and future. Journal of Public Transportation 2009;10:41–56.
[2] OBIS Project. Optimising bike sharing in European cities. A handbook, 2011.
[3] Pucher J, Buehler R, Seinen M. Bicycling renaissance in North America? An update and re-appraisal of cycling trends and policies Transportation Research Part A 2011;45:451–475.
[4] Shaheen S, Guzman S, Zhang H. Bikesharing in Europe, the Americas, and Asia: past, present, and future. Transportation Research Record 2010;2143:159–167.
[5] Kaltenbrunner A, Meza R, Grivolla J, Codina J, Banchs R. Urban cycles and mobility patterns: exploring and predicting trends in a bicycle-based public transport system. Pervasive and Mobile Computing 2010;6:455–466.
[6] Vogel P, Mattfeld D. Modeling of repositioning activities in bike-sharing systems. In: 12th WCTR, Lisbon, Portugal, 2010. p. 1–13.
[7] Lin J-R, Yang T. Strategic design of public bicycle sharing systems with service level constraints. Transportation Research Part E 2011;47:284–294.
[8] Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D. The planning of cycle trips in the province of East Flanders. Omega 2011;39:209–213.
[9] Boctor F, Renaud J, Cornillier F. Trip packing in petrol stations replenishment. Omega 2011;39:86–98.
[10] Berbeglia G, Cordeau J-F, Gribkovskaia I, Laporte G. Static pickup and delivery problems: a classification scheme and survey. TOP 2007;15:1–31.
[11] Parragh SN, Doerner KF, Hartl RF. A survey on pickup and delivery problems. Part I: transportation between customers and depot. Journal für Betriebswirtschaft 2008;58:21–51.
[12] Parragh SN, Doerner KF, Hart lRF. A survey on pickup and delivery problems. Part II: transportation between pickup and delivery locations. Journal für Betriebswirtschaft 2008;58:81–117.
[13] Battarra M, Cordeau J-F, Iori M. Pickup and delivery problems for goods transportation. Technical Report, University of Modena and Reggio Emilia, Italy; 2012.
[14] Doerner K, Salazar-González J-J. Pickup and delivery routing problems for people transportation. Technical Report, University of La Laguna, Spain; 2012.
[15] Hernández-Pérez H, Salazar-González J-J. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics 2004;145:126–139.
[16] Hernández-Pérez H, Salazar-González J-J. The one-commodity pickup-and-delivery traveling salesman problem: inequalities and algorithms. Networks 2007;50:258–272.
[17] Hernández-Pérez H, Salazar-González J-J. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. Transportation Science 2004;38:245–255.
[18] Hernández-Pérez H, Rodríguez-Martín I, Salazar-González J-J. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. Computers & Operations Research 2009;36:1639–1645.
[19] Wang F, Lim A, Xu Z. The one-commodity pickup and delivery travelling salesman problem on a path or a tree. Networks 2006;48:24–35.
[20] Martinović G, Aleksi I, Baumgartner A. Single-commodity vehicle routing problem with pickup and delivery service. Mathematical Problems in Engineering 2008 Article ID 697981.
[21] Zhao F, Li S, Sun J, Mei D. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. Computers & Industrial Engineering 2009;56:1642–1648.
[22] Hosny M, Mumford C. Solving the one-commodity pickup and delivery problem using an adaptive hybrid VNS/SA approach. In: Schaefer R, Cotta C, Kolodziej J, Rudolph G, editors. Parallel problem solving from nature, PPSN XI. Lecture notes in computer science, vol. 6239. Berlin: Springer; 2010. p. 189–198.
[23] Mladenović N, Urošević D, Hanafi S, Ilić A. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. European Journal of Operational Research 2012;220:270–285.
[24] Shi X, Zhao F, Gong Y. Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem. In: IEEE international conference on intelligent computing and intelligent systems, vol. 1, 2009. p. 175–9.
[25] Benchimol M, Benchimol P, Chappert B, De La Taille A, Laroche F, Meunier F, et al. Balancing the stations of a self service "Bike Hire" system. RAIRO—Operations Research 2011;45:37–61.
[26] Chemla D, Meunier F, Wolfler Calvo R. Bike sharing systems: solving the static rebalancing problem. Discrete Optimization 2013;10:120–146.
[27] Raviv T, Tzur M, Forma IA. Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal of Transportation and Logistics 2013;2:187–229.
[28] Contardo C, Morency C, Rousseau L-M. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, CIRRELT; 2012.
[29] Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 2006;34:209–219.
[30] Gutin G, Punnen A, editors. The traveling salesman and its variations. Dordrecht: Kluwer; 2002.
[31] Miller C, Tucker A, Zemlin R. Integer programming formulations and traveling salesman problems. Journal of the Association of Computing Machinery 1960;7:326–329.
[32] Baldacci R, Hadjiconstantinou E, Mingozzi A. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Operations Research 2004;52:723–738.
[33] Ascheuer N, Fischetti M, Grötschel M. A polyhedral study of the asymmetric travelling salesman problem with time windows. Networks 2000;36:69–79.
[34] Toth P, Vigo D. The vehicle routing problem. SIAM monographs on discrete mathematics and applications, Philadelphia, USA. 2002.