An Exact Algorithm for the Static Rebalancing Problem arising in Bicycle Sharing Systems

Güneş Erdoğan[1,*], Maria Battarra[2], and Roberto Wolfler Calvo[3]

[1] School of Management, University of Bath, East Building, BA2 7AY, UK,
G.Erdogan@bath.ac.uk

[2] School of Mathematics, University of Southampton, Highfield, Southampton, SO17 1BJ, UK,
M.Battarra@soton.ac.uk

[3] LIPN,Université de Paris Nord, 99, Avenue J.-B. Clément, 93430 Villetaneuse, France,
Roberto.Wolfler@lipn.univ-paris13.fr

[*] Corresponding author. Tel: +44 1225 386153. Fax: +44 1225 386473

February 23, 2015

Abstract

Bicycle sharing systems can significantly reduce traffic, pollution, and the need for parking spaces in city centers. One of the keys to success for a bicycle sharing system is the efficiency of rebalancing operations, where the number of bicycles in each station has to be restored to its target value by a truck through pickup and delivery operations. *The Static Bicycle Rebalancing Problem* aims to determine a minimum cost sequence of stations to be visited by a single vehicle as well as the amount of bicycles to be collected or delivered at each station. Multiple visits to a station are allowed, as well as using stations as temporary storage. This paper presents an exact algorithm for the problem and results of computational tests on benchmark instances from the literature. The computational experiments show that instances with up to 60 stations can be solved to optimality within 2 hours of computing time.

Keywords: Bicycle sharing systems, pickup and delivery, multiple visits, branch-and-cut.

1. Introduction

Urban transportation is a critical issue in large cities due to the dramatic increase and agglomeration of citizens and services in city centers. Congestion is a frequent and major issue and the associated decision problems are highly complex. Since the Buchanan report (Buchanan 1963), civil engineers proposed many effective techniques to mitigate the effects of urbanization on traffic. Shared mobility systems offer one of the most promising solutions and have the potential to reduce congestion in urban areas. In particular, bicycle sharing systems proved to be an effective solution to solve the "last mile" problem (Liu et al. 2012).

De Maio (2009) surveys the development of bicycle sharing systems starting from the 60's, when Witte Fiesten (white bicycles) were first introduced in Amsterdam. Although this first generation of shared bicycles did not prove to be successful, the second generation of bicycles introduced in Denmark in the early 90's was more effective, due to stronger and dedicated bicycles and coin-based payment systems. The third generation was first introduced in 1996 at the University of

Portsmouth in the UK, where students could rent a bicycle using a smart card. This generation of bicycles proved to be a success, drastically reducing the number of thefts and damaged bicycles. It quickly became clear that IT tools capable of tracking bicycles, storing information about their usage and data about the users were necessary to improve the quality of service and decrease the rate of stolen/damaged bicycles. De Maio (2009) reports about 120 bicycle sharing systems running as of 2009, some of which consist of tens of thousands of bicycles. Impressive results in terms of increased bicycle usage, reduced CO_2 gas emissions, and consequent public health improvements are also reported. The efficiency of bicycle sharing systems largely depends on the effectiveness of operational strategies implemented by the network operators.

One of the key requirements identified by De Maio (2009) for the fourth generation of bicycle sharing systems is a good redistribution system. Despite initiatives aimed at promoting users to redistribute bicycles, the most common technique implemented to relocate bicycles from areas of high supply/low demand to areas of low supply/high demand is using trucks. The use of large trucks can be expensive and have a heavy CO_2 footprint, therefore many cities migrate to electric vehicles or start using vehicle routing optimization software to decrease the transportation costs and fuel consumption.

In this paper, we study the *Static Bicycle Rebalancing Problem* (SBRP), introduced by Benchimol et al. (2011). The SBRP aims to find a minimum cost route for a vehicle that starts and ends its service at a *depot*, and restores the inventory level at every bicycle station to its target value by picking up and delivering bicycles as necessary. The vehicle may visit any station more than once, and may use stations as temporary storage for bicycles (i.e., preemption). The term *static* refers to the assumption that the number of bicycles at each station is known in advance and does not change during the pickup and delivery operations, as opposed to *dynamic* problems in which the number of bicycles may change during the operations due to users renting and returning bicycles. The SBRP is of interest for many bicycle sharing systems that rebalance the stations during the night. Although the SBRP has been studied by Chemla et al. (2013b), the authors succeeded in providing a strong lower bound and an effective heuristic solution method, but not an exact algorithm. In this study, we propose an exact algorithm for the SBRP and present extensive computational results on benchmark instances from the literature.

The remainder of this paper is organized as follows. In Section 2, we provide a survey of the literature on bicycle rebalancing problems arising in bicycle sharing systems. In Section 3, we present the mathematical problem definition of the SBRP. In Section 4, elements of the exact algorithm for the SBRP are stated. In Section 5, we present a simple heuristic for generating upper bounds. In Section 6, we provide the results of our computational experiments. Finally, we give our concluding remarks in Section 7.

2. Literature Survey

There is an increasing interest in optimization problems arising in bicycle sharing systems. Among the problems studied are the integration of bicycle sharing systems with other transportation systems (Chow and Sayarshad 2014), the problem of reserving parking spaces in one-way vehicle sharing systems (Kaspi et al. 2014), the dynamics of bicycles usage during the day (Agatz et al. 2011), and the effects of fuel price variations on the bicycles usage (Smith and Kauermann

2011). Recently, bicycle rebalancing problems have received a significant amount of interest from the research community. Given the high degree of complexity, metaheuristics have been proposed to tackle larger instances and problems with multiple vehicles. We refer the interested reader to the heuristic papers by Rainer-Harbach et al. (2014), Papazek et al. (2013), Gaspero et al. (2013), and Schuijbroek et al. (2013). In what follows, we will focus on the studies about exact methods for bicycle rebalancing problems.

Nair and Miller-Hooks (2011) study the dynamic problem of rebalancing a shared mobility system by using stochastic programming, where the demand at each station is modeled using a set of scenarios. Their modeling approach uses chance constraints, which guarantee that a given percentage of scenarios will be satisfied by the redistribution plans. The objective is to minimize the redistribution costs. However, this model does not return any operational routing decisions. Contardo et al. (2012) solve the dynamic rebalancing problem that aims to minimize the overall unmet demand, using a flow formulation defined on a space-time network. The formulation is solved by means of a Benders decomposition embedded in a column generation framework. We refer the interested reader to Chemla et al. (2013a) for further reading on dynamic problems.

Raviv et al. (2013) study the multi vehicles static rebalancing problem, in which the objective is to simultaneously minimize the routing cost and the customer dissatisfaction. The latter is modeled using a piecewise linear convex function of the number of bicycles at stations. The customer dissatisfaction function for a station attains its maximum when the station is full (the customers would not be able to return their bicycles) or empty (the customers would not be able to rent a bicycle). It is assumed that stations are visited at most once, service times are taken into account and the overall trip duration of each vehicle is bounded. The authors present two mathematical formulations, dominance rules, and valid inequalities. Their formulations are tested on artificial instances with up to 60 stations and one or two vehicles. The authors also introduce instances inspired by the Capital Bikeshare in Washington DC with up to 104 stations. Both formulations struggle to solve instances with two vehicles with respect to a single vehicle. Moreover, the authors point out that the constraint of a single visit is restrictive and may considerably limit the quality of the solutions achieved.

Chemla et al. (2013b) propose a mathematical formulation for the SBRP defined over a *time-expanded graph*, in which each station is replicated as many times as an upper bound on the number of visits possible to the station. The formulation uses four index variables and becomes intractable for medium-sized instances of SBRP. Therefore, the authors introduce two relaxations, the first of which uses two sets of two-index variables corresponding to the number of times each arc is traversed and the number of bicycles being carried on each arc, respectively. The second relaxation uses only variables representing the number of times an arc is traversed. The two relaxations are proven equivalent, but the linear relaxation of the second provides higher quality lower bounds, due to stronger capacity constraints. A Tabu Search algorithm is also developed and a set of benchmark instances is generated, with up to 100 stations to be visited. The authors report that the current number of stations visited in Paris by a vehicle with capacity $Q = 20$ bicycles is typically 50, with each station having capacity 30 bicycles. The gaps between the best upper bound solutions and lower bounds is roughly 2% for realistic-sized instances and up to 5% for instances with up to 100 stations.

Erdoğan et al. (2014) extend the single vehicle static rebalancing problem by assuming that the number of bicycles at a station after the repositioning should lie within a given interval rather than a specific target number, and a station can be visited at most once. The authors solve the problem exactly using two methods, a Benders decomposition based branch-and-cut algorithm and a traditional branch-and-cut algorithm. Drawing upon the similarity of the problem with the *One Commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP), they adapt valid inequalities studied in depth by Hernández-Pérez and Salazar-González (2004, 2007). Instances with up to 50 stations have been solved to optimality, and the Benders decomposition based branch-and-cut is observed to outperform the traditional branch-and-cut algorithm.

Dell'Amico et al. (2014) solve the multi vehicle static problem where the overall transportation cost is minimized and each station has to be visited exactly once. Four alternative mixed integer linear mathematical formulations are compared and inequalities are used to strengthen the formulations.

3. Problem definition

We now provide a mathematical definition of the SBRP. We are given a complete directed graph $G = (V, A)$. The vertex set $V = \{0, 1, \ldots, n\}$ consists of the depot (vertex 0, arrival and departure node for the vehicle) and the bicycle stations $V \setminus \{0\}$. The number of bicycles at a station $i \in \{1, \ldots, n\}$ is initially p_i, the target number of bicycles is q_i, and the capacity of station i is C_i. Note that the capacity is necessary because preemption is allowed. Therefore, stations that are initially *balanced* (i.e., $p_i = q_i$) could also be visited with the purpose of temporarily parking or collecting bicycles. Each arc (i, j) has an associated travel cost c_{ij}, which may represent the fuel consumption, the travel time, or the CO_2 emission. The vehicle can carry at most Q bicycles at a time. The objective is to minimize the overall solution cost, ensuring that the vehicle departs and arrives at the depot and the target number of bicycles is allocated to each station at the end of the tour.

The SBRP has been proven to be NP-hard by Benchimol et al. (2011), as well as its special cases for complete graphs and bipartite graphs with unit costs. The authors also propose a 9.5-approximation algorithm for the general case, a 2-approximation algorithm for the special case with a complete graph and unit costs, as well as lower bounding techniques. Chemla et al. (2013b) demonstrate that the problem of verifying if a sequence of vertices starting and ending at the depot for an instance of SBRP represents a feasible solution is a search problem (i.e., a polynomial time algorithm is capable of finding a feasible assignment of bicycles on the vehicle for each arc traveled, if any feasible assignment exists). On the other hand, if the only information available to represent a solution is the number of times each arc in the network is traversed, the problem of determining if this information corresponds to a feasible solution is NP-complete.

We now present a simplified version of the second relaxation of SBRP by Chemla et al. (2013b), which will serve as a stepping stone to a complete formulation. The authors define the imbalance of station $i \in V$ as $d_i = p_i - q_i$, the net imbalance of a subset of stations $S \subseteq V \setminus \{0\}$ as $d(S) = \sum_{i \in S} d_i$, and introduce a function $\mu(S)$ that is equal to 1 if there is at least one station $i \in S$ such that $|d_i| > 0$, and 0 otherwise. Based on these definitions, they use two separate sets of constraints for connectivity and capacity. These two sets of constraints can be expressed as a

single set by defining and using a single function to compute the minimal vehicle requirement of station set $S \subseteq V \setminus \{0\}$ as

$$r(S) = \max\{\lceil |d(S)|/Q \rceil, \mu(S)\}. \tag{1}$$

Furthermore, we share the definitions of $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$. Let x_{ij} be equal to the number of times arc $(i,j) \in A$ is traversed. We write $x(S)$ to denote $\sum_{(i,j)\in S} x_{ij}$. Chemla et al. (2013b) prove that an arc between two stations with both positive or both negative imbalance values need not be traversed more than once. When the travel costs respect the triangle inequality, we observe that for arcs $(i,j) \in A : d_i > 0, d_j < 0$ or $d_i < 0, d_j > 0$, the maximum number of traversals is equal to the minimum of the absolute values of the imbalances (i.e., at least one bicycle is transported at each pass). Hence, we denote the maximum number of times an arc can be traversed as

$$u_{ij} = \begin{cases} \min\{|d_i|, |d_j|\} & d_i > 0, d_j < 0 \text{ or } d_i < 0, d_j > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

The resulting relaxation is:

(SBRP-R)

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{3}$$

$$\text{subject to} \quad x(\delta^-(i)) = x(\delta^+(i)) \qquad (i \in V), \tag{4}$$

$$x(\delta^+(0)) \geq \max\{\lceil |d_0|/Q \rceil, 1\}, \tag{5}$$

$$x(\delta^+(S)) \geq r(S) \qquad (S \subseteq V \setminus \{0\}), \tag{6}$$

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

The objective function (3) minimizes the total travel cost. Constraint set (4) are the well-known flow conservation constraints. Constraint set (5) sets the minimum number of trips out of the depot. Constraint set (6) imposes the lower bounds for the number of times the vehicle must leave set S. Finally, constraint set (7) states the integrality constraints as well as the upper bounds on the variables.

There are two shortcomings of (SBRP-R) that prevent it from being a complete formulation. Firstly, as stated in the previous section, proving that an integral solution for (SBRP-R) is feasible for the SBRP is an NP-complete decision problem. Secondly, There is no straightforward method to separate an integer solution that is found to be infeasible for the SBRP. To observe this, assume that we have such a solution x^*. To separate this solution, the following constraint should be added to the constraint set.

$$\sum_{(i,j)\in A} |x_{ij} - x^*_{ij}| \geq 1. \tag{8}$$

Note that (8) is nonlinear, and there is no efficient method to linearize it for general integer variables. For binary variables, it is possible to linearize it by using $|x_{ij} - x^*_{ij}| = x_{ij}$ for $x^*_{ij} = 0$, and $|x_{ij} - x^*_{ij}| = 1 - x_{ij}$ for $x^*_{ij} = 1$. To utilize this linearization, we use the following transformation that converts general integer variables with finite upper bounds into a set of binary variables:

$$x_{ij} = \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk}, \tag{9}$$

where $y_{ijk} \in \{0,1\}, \quad \forall (i,j) \in A, k \in \{0, ..., \lfloor log_2(u_{ij}) \rfloor\}$. Note that for $u_{ij} : log_2(u_{ij}) = \lfloor log_2(u_{ij}) \rfloor$, the transformation allows for $\sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk} > u_{ij}$, but the upper bounds u_{ij} are provided to speed up the algorithm rather than feasibility.

Given an integer solution $\mathbf{y}^*$, let us define $I_0(\mathbf{y}^*) = \{(i,j,k) : y^*_{ijk} = 0\}$, i.e. the set of indices for the variables equal to 0. Similarly, we define $I_1(\mathbf{y}^*) = \{(i,j,k) : y^*_{ijk} = 1\}$, i.e. the set of indices for the variables equal to 1. Finally, we define the infeasibility indicator function $I_f(\mathbf{y}^*)$ that returns 1 if $\mathbf{y}^*$ is an integer solution that is infeasible for the SBRP, and 0 otherwise. Based on these definitions, it is possible to separate an integer solution that is infeasible for the SBRP with a *combinatorial Benders' cut* (Codato and Fischetti 2006):

$$\sum_{(i,j,k) \in I_0(\mathbf{y}^*)} y_{ijk} + \sum_{(i,j,k) \in I_1(\mathbf{y}^*)} (1 - y_{ijk}) \geq 1, \quad (\mathbf{y}^* : I_f(\mathbf{y}^*) = 1). \tag{10}$$

A similar approach has been independently developed and used by Chen et al. (2014), to linearize chance constraints for a model that optimizes road network daily maintenance operations with stochastic service and travel times. We are now ready to state the complete formulation for the SBRP.

(SBRP1)

$$\text{minimize} \quad \sum_{(i,j) \in A} c_{ij} \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk} \tag{11}$$

$$\text{subject to} \quad \sum_{j:(j,i) \in \delta^-(i)} \sum_{k=0}^{\lfloor log_2(u_{ji}) \rfloor} 2^k y_{jik} = \sum_{j:(i,j) \in \delta^+(i)} \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk} \qquad (i \in V), \tag{12}$$

$$\sum_{j:(0,j) \in \delta^+(\{0\})} \sum_{k=0}^{\lfloor log_2(u_{0j}) \rfloor} 2^k y_{0jk} \geq \max\{\lceil |d_0|/Q \rceil, 1\}, \tag{13}$$

$$\sum_{j:(i,j) \in \delta^+(S)} \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk} \geq r(S) \qquad (S \subseteq V \setminus \{0\}), \tag{14}$$

$$\sum_{(i,j,k) \in I_0(\mathbf{y}^*)} y_{ijk} + \sum_{(i,j,k) \in I_1(\mathbf{y}^*)} (1 - y_{ijk}) \geq 1 \qquad (\mathbf{y}^* : I_f(\mathbf{y}^*) = 1), \tag{15}$$

$$y_{ijk} \in \{0,1\} \quad ((i,j) \in A, k \in \{0, ..., \lfloor log_2(u_{ij}) \rfloor\}). \tag{16}$$

As in SBRP-R, the objective function (11) minimizes the total travel cost, constraint set (12) are the flow conservation constraints, constraint set (13) states the minimum number of trips out of the depot, and constraint set (14) imposes the lower bounds for the number of times the vehicle must leave S. Constraint set (15) separates integer feasible solutions that are not feasible for the SBRP. Finally, (16) state that all variables y_{ijk} are binary. To separate constraint set (14), we have used the separation algorithms described in Hernández-Pérez and Salazar-González (2007) for the 1-PDTSP. The separation algorithm for (15) is described in detail in the next section.

4. Separation algorithms

In this section, we provide the algorithm for separating the combinatorial Benders' cuts, and propose a modification of the separation algorithm that allows for the solution of the SBRP in which preemption is not allowed.

4.1. Separation algorithm for combinatorial Benders' cuts

The optimal solution of SBRP1 for the test instance n20q10C from the benchmark instance set of Chemla et al. (2013b) is depicted in Figure 1, where the number of arcs from $i \in V$ to $j \in V$ is computed as $x_{ij} = \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y_{ijk}$. Next to each vertex are the initial and target inventory levels corresponding to the vertex. Vertices 2, 10, 12, and 16 have an indegree of 2, whereas the rest of the vertices are visited only once. As stated in the previous section, checking if this solution corresponds to a feasible solution for the SBRP is an NP-complete decision problem. We now provide an algorithm for checking the feasibility of an integer feasible solution of SBRP1 for SBRP.

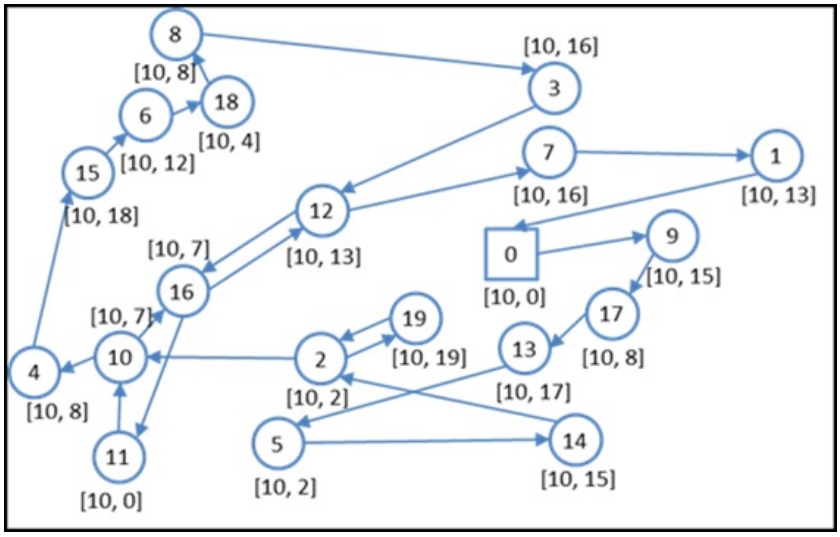

Figure 1: Optimal solution of the formulation SBRP1 for the instance n20q10C

The algorithm is based on two observations. Firstly, any solution for the SBRP that traverses the arcs of a solution of formulation SBRP1 corresponds to an Eulerian circuit starting and ending at the depot, on the graph defined by x_{ij}. Secondly, an Eulerian circuit starting and ending at the

depot can be represented as a time-extended network, where the time dimension of the vertices are defined by the order they are visited in. The feasibility of the SBRP solution can be checked by solving a flow problem on this network. Based on these observations, we can conclude that an exact separation algorithm for constraint set (15) should enumerate (implicitly or explicitly) all Eulerian circuits for an SBRP1 solution and check the feasibility of each Eulerian circuit by solving a flow problem on the corresponding time-extended network. A solution is declared feasible if at least one Eulerian circuit it contains has a feasible solution for the corresponding flow problem. Otherwise, the solution can be separated by adding an inequality from the constraint set (15).

We first present the method of constructing the time-extended network for the flow problem for a given Eulerian circuit $\bar{C} = \{0, v_1, v_2, ..., 0\}$. The reader can refer to the examples depicted in Figure 2 corresponding to the SBRP1 solution in Figure 1. In Figure 2, there is one graph for every Eulerian circuit corresponding to the SBRP1 solution in Figure 1. In each graph, the vertices are aligned to their indices presented on the left. A separate flow problem will be solved for each one of these graphs, and the feasibility of only one will suffice to prove feasibility of the solution. Denote the time-extended network of each graph in Figure 2 as $\hat{G} = (\hat{V}, \hat{A}_1 \cup \hat{A}_2)$. For the k^{th} vertex in $\bar{C}$, place a vertex j in $\hat{V}$, denote its *parent vertex* as $\pi(j)$, and its sequence number as $\sigma(j) = k$. If a vertex $i \in V$ occurs only once in $\bar{C}$, set the supply/demand value of the corresponding vertex in $j \in \hat{V}$ to $b_j = p_i - q_i$. If vertex $i \in V$ occurs twice or more in the circuit, set the supply/demand value of the first occurrence to $b_j = p_i$, the last occurrence to $b_k = -q_i$, and the remaining occurrences to $b_l = 0$. Add an arc (i, j) of capacity Q to $\hat{A}_1$, $\forall i, j \in \hat{V} : \sigma(j) = \sigma(i)+1$. Finally, add an arc (i, j) of capacity $C_{\pi(i)}$ to $\hat{A}_2$, $\forall i, j \in \hat{V} : \pi(i) = \pi(j), \sigma(j) = \min_{k \in \hat{V}: \pi(k)=\pi(i), \sigma(k)>\sigma(i)} \sigma(k)$. Defining z_{ij} as the amount of flow on arc $(i, j) \in \hat{A}_1 \cup \hat{A}_2$, the resulting flow problem can be stated as:

(TEFP)

$$\text{minimize } 0 \tag{17}$$

$$\text{subject to} \quad z(\delta^+(i)) - z(\delta^-(i)) = b_i \qquad (i \in \hat{V}), \tag{18}$$

$$z_{ij} \leq Q \qquad ((i, j) \in \hat{A}_1), \tag{19}$$

$$z_{ij} \leq C_{\pi(i)} \qquad ((i, j) \in \hat{A}_2), \tag{20}$$

$$z_{ij} \geq 0 \qquad ((i, j) \in \hat{A}_1 \cup \hat{A}_2). \tag{21}$$

The objective function (17) is empty since TEFP is a feasibility problem. Constraint set (18) enforces the flow conservation for the bicycles at each station. Constraint sets (19) and (20) are the capacity constraints for the vehicle and stations, respectively. Finally, constraint set (21) consists of the nonnegativity constraints. Note that the size of the network $\hat{G}$ is determined by the supply/demand values as well as the number of vertices, which may be of exponential size, inhibiting a polynomial time complexity.

The flow conservation constraints of SBRP1 ensure that an Eulerian circuit exists for every integral solution. However, there may exist an exponential number of Eulerian circuits in a given digraph (Tutte 2001). We now provide the pseudocode of a search algorithm that solves the separation problem, by enumerating Eulerian circuits and testing their feasibility using TEFP.

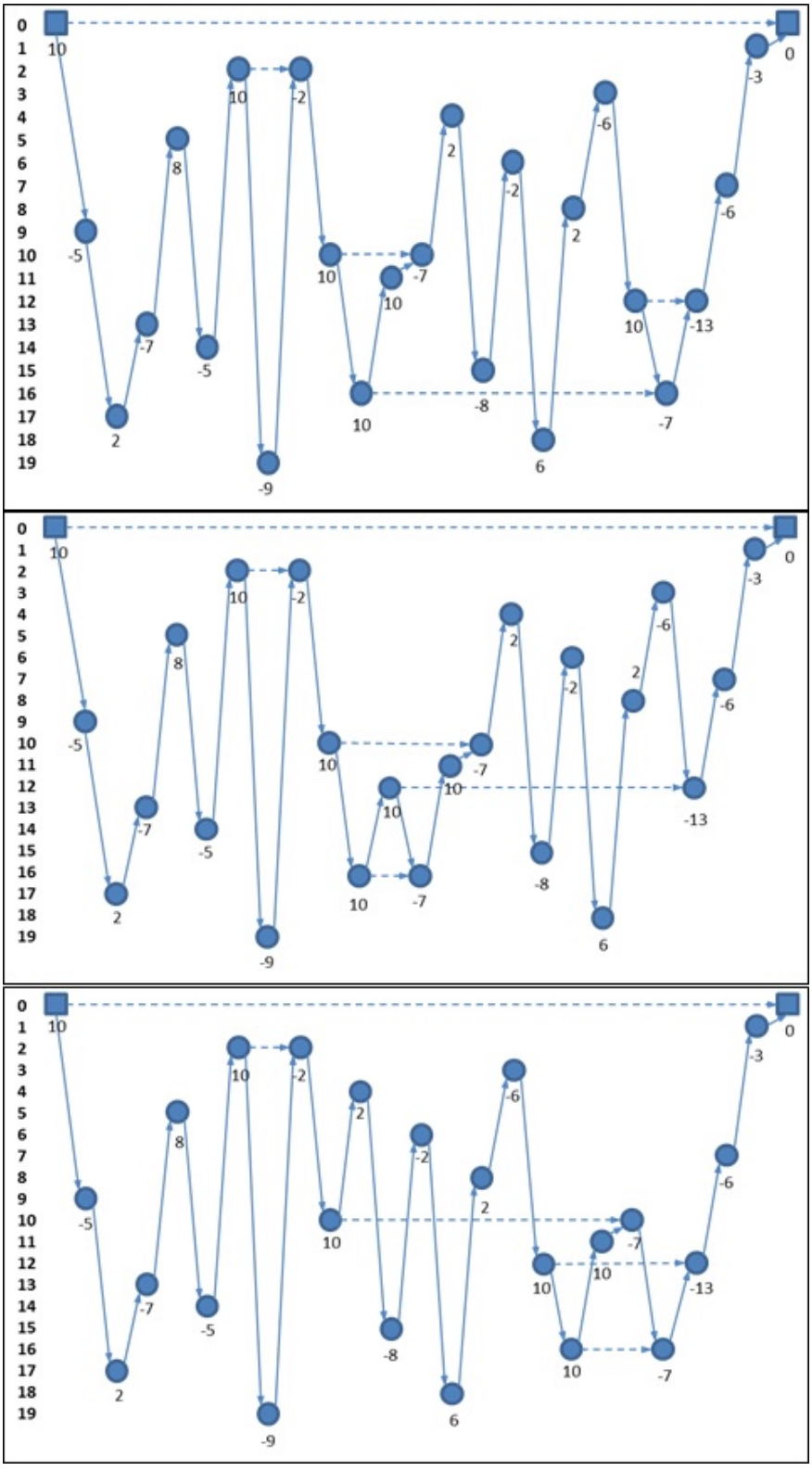

Figure 2: Time-extended flow problems for the instance n20q10C

Separation(vertex i, counter l, list $\bar{C}$, solution $\mathbf{x}^*$)

If *feasible* = true

Return

End If

If $l = 0$

$i = 0$, $\bar{C} = \{(0,0)\}$

$x^*_{ij} = \sum_{k=0}^{\lfloor log_2(u_{ij}) \rfloor} 2^k y^*_{ijk} \quad \forall (i,j) \in A$

End If

If $\sum_{(i,j)\in A} x^*_{ij} = 0$

Solve the TEFP instance corresponding to the Eulerian circuit $\bar{C}$.

If the TEFP instance has a feasible solution

feasible = true

Return

Else

Check the reachability of every vertex from i on the graph induced by $\mathbf{x}^*$.

If $\exists j \in V : \sum_{k:(k,j)\in A} x^*_{kj} > 0$ that is unreachable from i

Return

End If

For $j : (i,j) \in A$ // branching

If $x^*_{ij} > 0$

$x^*_{ij} = x^*_{ij} - 1$

$\bar{C} = \bar{C} \cup (j, l+1)$

Separation(j, $l+1$, $\bar{C}$, $\mathbf{x}^*$)

$x^*_{ij} = x^*_{ij} + 1$

$\bar{C} = \bar{C} \setminus (j, l+1)$

End If

End For

Return

The global variable $feasible$ should be initialized as false before the separation algorithm, and the algorithm should be invoked as **Separation**$(0, 0, \emptyset, \mathbf{x}^*)$. The first step of the algorithm is to set the beginning of the Eulerian circuit as the depot, and compute the number of times each arc is traversed. Starting at the depot, the algorithm proceeds as a depth-first search, branching on the arcs leaving the current vertex. If a vertex with a positive indegree becomes unreachable from the current vertex i, the node of the search tree is fathomed due to infeasibility. Every time the algorithm traverses all the arcs (i.e. finds a Eulerian circuit), an instance of TEFP is solved and the search algorithm terminates upon finding a feasible solution. If no feasible TEFP solutions are found, the solution is declared infeasible and is separated with a combinatorial Benders' cut. Else, the solution of the TEFP is recorded as the certificate of feasibility.

We would like to emphasize that the branching order is important for the efficiency of the separation algorithm. In our computational experiments, we have found it useful to solve a network flow problem on $G = (V, A)$, where arc capacities given by $Q\,\mathbf{x}^*$ and vertex demands are given by

$d(i)$, $\forall i \in V$, to identify the feasible flow patterns. Based on the solution of the flow problem, we explore the branches in decreasing order of flow from supply vertices ($d_i > 0$), and in decreasing order of the destination vertices' supply values from transshipment and demand vertices ($d_i \leq 0$).

4.2. Separation algorithm for the nonpreemptive SBRP

The benefits of preemptive pickup and delivery operations with respect to nonpreemptive operations are of particular interest to practitioners and academics alike. Although it is straightforward to see the advantage of using temporary storage, the trade-off between the monetary benefit and the added complexity of operations is hard to judge. In what follows, we describe a separation algorithm that identifies integral solutions that are not feasible with respect to nonpreemptive operations.

Given an Eulerian circuit $\bar{C} = \{0, v_1, v_2, ..., 0\}$, construct the time-extended network $\hat{G} = (\hat{V}, \hat{A}_1 \cup \hat{A}_2)$ as described above, along with the supply/demand values, and the capacities of all arcs. The nonpreemptive constraint requires the vertices with an imbalance of 0 not to be visited, the vertices with negative imbalance to have monotonically increasing amounts of inventory, and the vertices with positive imbalance to have monotonically decreasing amounts of inventory. Consequently, the following constraints should be added to SBRP1:

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 0 \qquad ((i,j) \in \hat{A}_2, d_i = 0). \tag{22}$$

Let us denote the formulation obtained by adding constraints (22) to SBRP1 as SBRP2. In addition, the following constraints should be added to TEFP:

$$z_{ij} \geq p_i \qquad ((i,j) \in \hat{A}_2, d_{\pi(i)} < 0), \tag{23}$$

$$z_{jk} \geq z_{ij} \qquad ((i,j),(j,k) \in \hat{A}_2, d_{\pi(i)} < 0), \tag{24}$$

$$z_{ij} \leq p_i \qquad ((i,j) \in \hat{A}_2, d_{\pi(i)} > 0), \tag{25}$$

$$z_{jk} \leq z_{ij} \qquad ((i,j),(j,k) \in \hat{A}_2, d_{\pi(i)} > 0). \tag{26}$$

Let us denote the formulation obtained by adding constraints (23)-(26) to TEFP as TEFP2. We now prove that although the constraints (24) and (26) impair the integrality property of TEFP2, this formulation can still be solved in polynomial time, since the existence of a fractional feasible solution implies the existence of an integer feasible solution.

Proposition 1: Every fractional feasible solution to TEFP2 corresponds to an integral feasible solution.

Proof: Consider the graph $\hat{G} = (\hat{V}, \hat{A}_1 \cup \hat{A}_2)$ defined in the previous subsection. To ensure that the nonpreemptive constraint is respected, we modify the graph by adding a copy of the vertices $j \in \hat{V}$ that correspond to the visits other than the first or the last. The modification is depicted in Figure 3, where we assume that more than two visits are taking place at station $\pi(j) \in V$. We name the copies as j_1 and j_2, and connect the associated arcs in $\hat{A}_1$ to j_1 and the associated arcs in

$\hat{A}_2$ to j_2. For $d_{\pi(j)} > 0$, we place an arc from j_2 to j_1 that only allows flow out of the vertex $\pi(j)$. Similarly, for $d_{\pi(j)} < 0$, we place an arc from j_1 to j_2 that only allows flow into the vertex $\pi(j)$. We define the set of such arcs as $\hat{A}_3$. The corresponding time-extended network flow problem has a larger number of vertices and arcs, respects the nonpreemptive constraint, but benefits from the integrality property. Consider the flow variable for an arc $(j_1, j_2) \in \hat{A}_3 : d_{\pi(j_1)} < 0$. This variable is present only in two flow conservation constraints and a nonnegativity constraint:

$$z(\delta^+(j_1) \setminus \{(j_1, j_2)\}) + z_{j_1,j_2} - z(\delta^-(j_1)) = 0, \tag{27}$$

$$z(\delta^+(j_2)) - z(\delta^-(j_2) \setminus \{(j_1, j_2)\}) - z_{j_1,j_2} = 0, \tag{28}$$

$$z_{j_1,j_2} \geq 0. \tag{29}$$

Applying Fourier-Motzkin elimination (Dantzig and Eaves 1973) on the variable z_{j_1,j_2} yields:

$$z(\delta^+(j_2)) + z(\delta^+(j_1) \setminus \{(j_1, j_2)\}) = z(\delta^-(j_1)) + z(\delta^-(j_2) \setminus \{(j_1, j_2)\}), \tag{30}$$

$$z(\delta^+(j_1) \setminus \{(j_1, j_2)\}) \leq z(\delta^-(j_1)), \tag{31}$$

$$z(\delta^+(j_2)) \geq z(\delta^-(j_2) \setminus \{(j_1, j_2)\}.) \tag{32}$$

It can be easily observed that (30) and (31) imply (32), which renders (32) redundant. Analysis of (30) and (31) shows that the Fourier-Motzkin elimination merges j_1 and j_2 back into j, since (30) corresponds to the original flow conservation constraint for vertex j. In addition, (31) corresponds to (24). The analysis for $(j_2, j_1) \in \hat{A}_3 : d_{\pi(j_1)} > 0$ yields a similar result, showing that (26) is implied by the flow conservation and nonnegativity constraints. Due to the equivalence of the formulations, any fractional feasible solution for TEFP2 corresponds to an integral feasible solution. Hence, if the separation algorithm is executed on the integer solutions of SBRP2 by solving TEFP2 as a continuous problem, it successfully identifies integer solutions that do not satisfy the nonpreemptive constraint. ■

We would like to note that TEFP2 should be solved with integrality constraints for the final result of the branch-and-cut algorithm, to determine the amounts of pickup and delivery at each station.

5. Constructive heuristic

The separation algorithms presented in the previous section perform well for near-optimal solutions, but may perform poorly for low quality solutions. In particular, for "star-shaped" solutions, in which the degree of a vertex is much higher than the others, the branching order may not offer sufficient guidance to find a feasible solution: many infeasible solutions should be enumerated before the algorithm can conclude feasibility. To avoid this stalling behavior, we now present a greedy constructive heuristic that is slightly different from the greedy heuristic of Chemla et al. (2013b), which prioritizes exhausting the supply/demand of vertices.

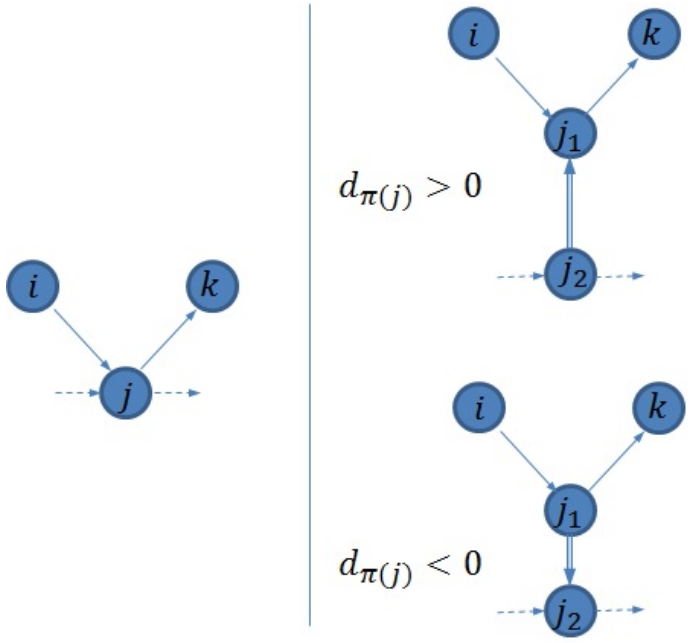

Figure 3: The transformation of the time-extended graph for nonpreemptive SBRP

Starting from an empty solution, the constructive heuristic picks up/delivers the maximal possible amount at each vertex. If the vehicle is full, the next vertex is chosen as the closest vertex with negative imbalance. Else if the vehicle is empty, the closest vertex with supply becomes the next vertex to be visited. Else, the closest vertex with a nonnegative imbalance is visited. The heuristic successfully overcomes the stalling problem at the cost of less than 0.01 seconds of CPU time, and its result is feasible for both SBRP and nonpreemptive SBRP.

Constructive Heuristic

$x^*_{ij} = 0 \quad \forall (i,j) \in A$
$\hat{d}_j = d_j \quad \forall j \in V$ // temporary supply/demand values
$r = 0$ // quantity on board
$i = 0$ // current vertex
While $\sum_{j \in V} |\hat{d}_j| > 0$
    // pickup/delivery at current vertex
    If $\hat{d}_i < 0$
        If $r + \hat{d}_i \geq 0$
            $r = r + \hat{d}_i$
            $\hat{d}_i = 0$
        Else
            $\hat{d}_i = \hat{d}_i + r$
            $r = 0$
    If $\hat{d}_i > 0$
        If $r + \hat{d}_i \leq Q$
            $r = r + \hat{d}_i$

$\hat{d}_i = 0$

Else

$\hat{d}_i = \hat{d}_i - (Q - r)$

$r = Q$

// determine the next vertex

If $\sum_{j \in V} |\hat{d}_j| = 0$

If $i = 0$

Return

Else

$k = 0$

Else If $r = Q$

$k = \text{argmin}_{\ j:j \in V, \hat{d}_j < 0} c_{ij}$

Else If $r = 0$

$k = \text{argmin}_{\ j:j \in V, \hat{d}_j > 0} c_{ij}$

Else

$k = \text{argmin}_{\ j:j \in V, |\hat{d}_j| > 0} c_{ij}$

$x^*_{ik} = x^*_{ik} + 1$

$i = k$

Return

---

6. Computational Results

The algorithm described in the previous section has been implemented using C++ and CPLEX 12.5, on IRIDIS 4 computing cluster consisting of 2.6 GHz cores with 4 GB of memory per core. The result of the constructive heuristic was fed into CPLEX as a starting solution. The branch-and-cut algorithm of CPLEX was restricted to run on a single core, to enable a better analysis of the algorithm by excluding the parallel speedup.

The instances used are identical to those of Chemla et al. (2013b). We replicate the details of the instance generation scheme for the sake of simplicity. Chemla et al. (2013b) adapt the instances of Hernández-Pérez and Salazar-González (2004) for the 1-PDTSP. The original instances consist of Euclidean coordinates for the vertices, and an integer demand value $\tilde{d}_i \in \{-10, ..., 10\}$ for every vertex $i \in V$. The resulting instances have $p_i = \alpha \times 10$, $q_i = \alpha \times (10 + \tilde{d}_i)$, and $C_i = \alpha \times 20$. Chemla et al. (2013b) have experimented with $\alpha = 1$ and $\alpha = 3$, corresponding to small and large amounts to be relocated. The original instance set consists of 10 instances for each number of vertices $|V|$ and vehicle capacity Q.

The computational experiments have been carried out on sets of instances with $|V| \in \{20, 30, 40, 50, 60\}$. Due to space considerations, aggregate results are presented for $\alpha = 1$ and $\alpha = 3$ in Tables 1 and 2, respectively, where every row corresponds to the average results for 10 instances. Detailed results can be obtained from the authors. The column heading "Average Heuristic Gap" refers to the mean percentage deviation of the result of the constructive heuristic from the best known lower bound. Similarly, the column heading "Average Optimality Gap" refers to the mean percentage deviation of the best result of the branch-and-cut algorithm from the best

known lower bound. It can be observed from Table 2 that the instances are harder for $\alpha = 3$, in particular for $|V| \geq 50$. This is due to the fact that more bicycles to carry result in more arcs traversed, and more Eulerian circuits. For $\alpha = 1$, we were able to solve all but 1 instance with 40 vertices, 7 instances with 50 vertices, and 18 instances with 60 vertices. For $\alpha = 3$, we were able to solve all instances up to 40 vertices, 69 out of 90 instances with 50 vertices, and 60 out of 90 instances with 60 vertices. The constructive heuristic consistently provided initial solutions that are 40% to 50% away from the best known lower bound, for both. We conclude that the computational reach of the algorithm is about 60 vertices.

We have also solved the same set of instances for the nonpreemptive SBRP. The aggregate results are presented in Tables 3 and 4, respectively. The complexity increases significantly for $|V| \geq 40$. In addition, the separation algorithm could not conclude feasibility or infeasibility for 3 instances with $\alpha = 1$, and 20 instances with $\alpha = 3$. We deduce that the computational reach for the nonpreemptive SBRP is approximately 40 vertices. The performance of the constructive heuristic is very similar for both SBRP and nonpreemptive SBRP. We have also compared the results of the SBRP and nonpreemptive SBRP, for $|V| \in \{20, 30, 40\}$. The results demonstrate that the added value of preemption is around 0.6%.

We have also tested our algorithm on a subset of the real world instances used in the study of Dell'Amico et al. (2014), specifically the first 50 instances which involve up to 59 stations. This set of experiments have been performed on another computer with an Intel i7 CPU running at 3.60 Ghz and 8 GB of RAM. The software parameters are identical to the previous set of experiments. The demand values for this set of instances were not balanced (i.e. did not sum up to 0), so we have balanced the demand values by setting the demand of the depot to $d_0 = -\sum_{i \in V \setminus \{0\}} d_i$. Finally, we have determined the initial level of inventory at the stations as $\max_{i \in V} |d_i|$. The details of our results are provided in Tables 5 - 8. The performances of our algorithms are slightly better for this data set, and our observations for the previous data set are confirmed.

As a final note, we observe that the longer runs for both variants are for the instances for which CPLEX fails to find a high quality solution in the early stages of the branch-and-cut algorithm. Hence, tight upper bounds supplied by a sophisticated metaheuristic algorithm may improve the performance of the exact algorithm.

7. Conclusion

In this study, we have studied the SBRP and provided an exact solution algorithm. The algorithm is based on an integer programming formulation, which is obtained through a variable transformation that enables to convert all general integer variables to binary variables. We have utilized combinatorial Benders' cuts to separate infeasible solutions from the feasible region. Separation of the combinatorial Benders' cuts was achieved through an enumerative search algorithm. The algorithm was computationally tested on two instance sets from the literature, and was found to be capable of solving instances with up to 60 stations. We have also provided an extension of the algorithm that can solve the nonpreemptive SBRP. The computational experiments demonstrated that the added value of preemption is about 0.6%, and the nonpreemptive SBRP is significantly harder to solve.

Table 1: Aggregate results for $\alpha = 1$

|V|	Q	Average Optimality Gap	Average Heuristic Gap	Average CPU time	Number of Instances Solved
20	10	0.00%	37.67%	0.35	10
20	15	0.00%	40.78%	0.30	10
20	20	0.00%	48.24%	0.13	10
20	25	0.00%	46.85%	0.15	10
20	30	0.00%	45.31%	1.96	10
20	35	0.00%	46.86%	1.12	10
20	40	0.00%	46.87%	1.22	10
20	45	0.00%	46.87%	1.13	10
20	1000	0.00%	46.87%	0.83	10
30	10	0.00%	42.83%	6.22	10
30	15	0.00%	39.25%	3.87	10
30	20	0.00%	34.58%	163.59	10
30	25	0.00%	36.53%	5.61	10
30	30	0.00%	33.99%	82.20	10
30	35	0.00%	29.79%	293.27	10
30	40	0.00%	31.05%	584.62	10
30	45	0.00%	30.89%	221.69	10
30	1000	0.00%	30.99%	190.20	10
40	10	0.00%	52.66%	124.80	10
40	15	0.00%	47.77%	25.55	10
40	20	0.00%	41.21%	14.72	10
40	25	0.03%	40.88%	723.88	9
40	30	0.00%	37.36%	36.56	10
40	35	0.00%	38.90%	38.66	10
40	40	0.00%	39.31%	70.65	10
40	45	0.00%	39.35%	74.28	10
40	1000	0.00%	39.31%	70.17	10
50	10	0.79%	56.80%	1198.48	9
50	15	0.43%	55.61%	1970.12	8
50	20	0.00%	49.44%	295.45	10
50	25	0.00%	52.51%	272.82	10
50	30	0.00%	46.62%	177.40	10
50	35	0.24%	39.01%	1461.09	8
50	40	0.01%	36.17%	1408.94	9
50	45	0.00%	38.98%	1221.33	10
50	1000	0.11%	37.51%	1909.76	9
60	10	1.24%	45.58%	3924.62	6
60	15	0.51%	49.70%	1957.50	8
60	20	0.00%	45.97%	1285.03	10
60	25	0.07%	51.68%	943.42	9
60	30	0.13%	41.00%	1252.65	9
60	35	0.13%	41.39%	1096.98	9
60	40	0.22%	40.48%	2607.91	8
60	45	0.15%	41.19%	2795.20	8
60	1000	0.18%	41.60%	2816.42	8

Table 2: Aggregate results for $\alpha = 3$

|V|	Q	Average Optimality Gap	Average Heuristic Gap	Average CPU time	Number of Instances Solved
20	10	0.00%	30.74%	0.48	10
20	15	0.00%	34.81%	0.32	10
20	20	0.00%	37.24%	0.37	10
20	25	0.00%	38.96%	0.39	10
20	30	0.00%	37.67%	0.35	10
20	35	0.00%	42.75%	0.33	10
20	40	0.00%	43.32%	0.28	10
20	45	0.00%	40.78%	0.31	10
20	1000	0.00%	46.87%	0.93	10
30	10	0.00%	32.31%	153.85	10
30	15	0.00%	34.09%	65.36	10
30	20	0.00%	40.82%	8.16	10
30	25	0.00%	35.87%	10.27	10
30	30	0.00%	41.83%	9.91	10
30	35	0.00%	41.92%	9.37	10
30	40	0.00%	42.82%	5.92	10
30	45	0.00%	38.89%	2.96	10
30	1000	0.00%	30.15%	221.30	10
40	10	0.00%	35.46%	235.42	10
40	15	0.00%	38.13%	28.83	10
40	20	0.00%	41.00%	62.22	10
40	25	0.00%	45.61%	177.39	10
40	30	0.00%	49.15%	108.31	10
40	35	0.00%	43.51%	304.70	10
40	40	0.00%	48.26%	21.68	10
40	45	0.00%	44.57%	25.74	10
40	1000	0.00%	39.13%	80.91	10
50	10	0.99%	37.37%	2693.41	7
50	15	0.00%	39.19%	1702.17	10
50	20	0.89%	49.35%	3085.39	7
50	25	0.59%	52.73%	2024.55	8
50	30	0.46%	51.40%	1345.21	9
50	35	1.27%	50.53%	4212.45	5
50	40	0.45%	49.30%	3545.41	6
50	45	0.23%	48.80%	2057.76	8
50	1000	0.10%	36.07%	1938.98	9
60	10	2.83%	37.56%	3718.02	6
60	15	2.73%	40.91%	2932.58	7
60	20	0.39%	46.72%	3772.60	7
60	25	2.63%	47.89%	3636.12	6
60	30	2.88%	45.63%	4702.35	6
60	35	2.42%	50.27%	4795.69	6
60	40	0.46%	53.08%	3829.99	7
60	45	0.26%	49.61%	2223.52	8
60	1000	0.18%	41.58%	2940.39	7

Table 3: Aggregate results for nonpreemptive SBRP, $\alpha = 1$

|V|	Q	Average Optimality Gap	Average Preemption Gap	Average Heuristic Gap	Average CPU time	Number of Instances Solved
20	10	0.00%	0.29%	37.24%	3.62	10
20	15	0.00%	1.05%	39.28%	9.50	10
20	20	0.00%	0.34%	47.74%	0.36	10
20	25	0.00%	0.70%	45.86%	0.62	10
20	30	0.00%	0.44%	44.46%	2.16	10
20	35	0.00%	1.14%	45.33%	89.56	10
20	40	0.00%	1.14%	45.34%	102.11	10
20	45	0.00%	1.14%	45.34%	94.56	10
20	1000	0.00%	1.14%	45.34%	93.04	10
30	10	0.00%	0.11%	41.67%	44.63	10
30	15	0.00%	0.04%	38.84%	4.45	10
30	20	0.00%	0.02%	33.96%	138.97	10
30	25	0.00%	0.05%	35.97%	4.01	10
30	30	0.00%	0.33%	33.01%	64.16	10
30	35	0.00%	0.46%	28.45%	640.71	10
30	40	0.00%	0.33%	29.19%	182.66	10
30	45	0.00%	0.33%	29.45%	145.70	10
30	1000	0.00%	0.33%	29.56%	125.40	10
40	10	0.39%	0.55%	48.91%	1538.48	8
40	15	0.00%	0.30%	44.13%	444.62	10
40	20	0.14%	0.61%	39.12%	1241.54	9
40	25	0.15%	0.86%	39.41%	1132.37	9
40	30	0.15%	0.85%	36.32%	831.15	9
40	35	0.15%	0.77%	37.87%	798.08	9
40	40	0.16%	0.86%	38.23%	855.00	9
40	45	0.16%	0.86%	38.23%	859.48	9
40	1000	0.16%	0.86%	38.23%	892.54	9
50	10	0.12%	N/A	52.46%	2211.35	8
50	15	5.26%	N/A	48.59%	3693.62	6
50	20	0.13%	N/A	43.44%	1288.81	9
50	25	0.21%	N/A	47.60%	1326.72	9
50	30	0.37%	N/A	44.03%	2719.31	8
50	35	0.60%	N/A	36.84%	3265.79	7
50	40	0.32%	N/A	34.18%	2115.63	8
50	45	0.35%	N/A	36.70%	2399.27	8
50	1000	0.84%	N/A	35.60%	4236.27	6
60	10	8.45%	N/A	45.97%	4272.51	5
60	15	7.53%	N/A	50.17%	3725.42	6
60	20	0.17%	N/A	45.82%	2577.37	8
60	25	0.19%	N/A	51.38%	2307.77	7
60	30	0.75%	N/A	40.81%	3196.17	6
60	35	1.05%	N/A	41.06%	4126.38	5
60	40	1.13%	N/A	39.98%	4861.34	4
60	45	1.51%	N/A	40.93%	5127.30	3
60	1000	1.21%	N/A	39.23%	5757.87	2

Table 4: Aggregate results for nonpreemptive SBRP, $\alpha = 3$

|V|	Q	Average Optimality Gap	Average Preemption Gap	Average Heuristic Gap	Average CPU time	Number of Instances Solved
20	10	0.00%	0.00%	30.74%	0.62	10
20	15	0.00%	0.06%	34.73%	0.68	10
20	20	0.00%	0.08%	37.12%	0.67	10
20	25	0.00%	0.00%	38.96%	0.32	10
20	30	0.00%	0.29%	37.24%	3.33	10
20	35	0.00%	0.42%	42.18%	10.20	10
20	40	0.00%	0.52%	42.58%	8.29	10
20	45	0.00%	1.05%	39.28%	8.66	10
20	1000	0.00%	1.14%	45.34%	91.92	10
30	10	0.00%	0.01%	32.30%	185.85	10
30	15	0.00%	0.00%	34.09%	60.06	10
30	20	0.00%	0.00%	40.82%	4.94	10
30	25	0.00%	0.06%	35.80%	67.76	10
30	30	0.00%	0.11%	41.67%	44.64	10
30	35	0.00%	0.00%	41.92%	8.65	10
30	40	0.06%	0.28%	42.52%	723.65	9
30	45	0.00%	0.04%	38.84%	4.47	10
30	1000	0.00%	0.33%	29.56%	118.56	10
40	10	0.00%	0.01%	33.08%	1007.05	9
40	15	0.00%	0.00%	38.13%	97.74	10
40	20	0.11%	0.20%	40.14%	2186.83	7
40	25	0.02%	0.08%	44.17%	1618.70	8
40	30	0.39%	0.55%	48.91%	1538.21	8
40	35	0.42%	0.63%	43.21%	2333.29	7
40	40	0.22%	0.55%	47.77%	1178.96	9
40	45	0.00%	0.30%	44.13%	450.04	10
40	1000	0.16%	0.86%	38.23%	893.64	9
50	10	0.08%	N/A	38.20%	3461.00	6
50	15	0.00%	N/A	40.20%	2702.22	7
50	20	2.51%	N/A	50.59%	3151.67	6
50	25	1.77%	N/A	52.56%	2329.07	8
50	30	4.68%	N/A	51.79%	2300.22	8
50	35	1.55%	N/A	50.13%	4067.33	5
50	40	1.58%	N/A	48.95%	3529.59	7
50	45	5.26%	N/A	48.59%	3769.49	6
50	1000	0.85%	N/A	35.62%	4347.92	6
60	10	2.74%	N/A	36.39%	3851.62	6
60	15	0.48%	N/A	42.38%	3562.74	6
60	20	2.99%	N/A	48.30%	4638.22	5
60	25	13.54%	N/A	49.22%	3933.81	5
60	30	8.50%	N/A	45.99%	4347.23	5
60	35	4.63%	N/A	50.08%	4523.94	6
60	40	1.97%	N/A	52.96%	3534.82	6
60	45	0.39%	N/A	48.36%	3730.08	6
60	1000	1.19%	N/A	39.21%	5759.18	2

Table 5: Results for the real world data, preemptive SBRP, $\alpha = 1$

City	|V|	Q	Best known solution value	Heuristic solution value	Best known lower bound	Optimality Gap	Heuristic Gap	CPU time
Bari	13	30	14600	19000	14600	0.00%	30.14%	0.05
Bari	13	20	15700	19000	15700	0.00%	21.02%	0.03
Bari	13	10	20600	27900	20600	0.00%	35.44%	0.03
ReggioEmilia	14	30	16900	27600	16900	0.00%	63.31%	0.02
ReggioEmilia	14	20	23200	33700	23200	0.00%	45.26%	0.02
ReggioEmilia	14	10	32500	43500	32500	0.00%	33.85%	0.04
Bergamo	15	30	12600	18600	12600	0.00%	47.62%	0.04
Bergamo	15	20	12700	18600	12700	0.00%	46.46%	0.04
Bergamo	15	12	13500	18000	13500	0.00%	33.33%	0.07
Parma	15	30	29000	36300	29000	0.00%	25.17%	0.03
Parma	15	20	29000	36300	29000	0.00%	25.17%	0.03
Parma	15	10	32500	40500	32500	0.00%	24.62%	0.05
Treviso	18	30	29259	42001	29259	0.00%	43.55%	0.05
Treviso	18	20	29259	42001	29259	0.00%	43.55%	0.05
Treviso	18	10	31443	42001	31443	0.00%	33.58%	0.05
LaSpezia	20	30	20746	36096	20746	0.00%	73.99%	0.03
LaSpezia	20	20	20746	36096	20746	0.00%	73.99%	0.03
LaSpezia	20	10	22811	41250	22811	0.00%	80.83%	0.08
BuenosAires	21	30	71965	107385	71965	0.00%	49.22%	4.08
BuenosAires	21	20	85120	108596	85120	0.00%	27.58%	0.45
Ottawa	21	30	16202	23609	16202	0.00%	45.72%	0.04
Ottawa	21	20	16202	23609	16202	0.00%	45.72%	0.03
Ottawa	21	10	17370	23609	17370	0.00%	35.92%	0.06
SanAntonio	23	30	22982	36644	22982	0.00%	59.45%	0.07
SanAntonio	23	20	23887	36692	23887	0.00%	53.61%	0.08
SanAntonio	23	10	39553	53231	39553	0.00%	34.58%	0.30
Brescia	27	30	30300	44200	30300	0.00%	45.87%	0.24
Brescia	27	20	31100	47800	31100	0.00%	53.70%	0.39
Brescia	27	11	34900	56900	34900	0.00%	63.04%	0.47
Roma	28	30	61800	99600	61800	0.00%	61.17%	0.56
Roma	28	20	65600	71500	65600	0.00%	8.99%	1.65
Roma	28	18	68000	108600	68000	0.00%	59.71%	0.90
Madison	28	30	29246	46816	29246	0.00%	60.08%	0.05
Madison	28	20	29839	40463	29839	0.00%	35.60%	0.10
Madison	28	10	33627	49165	33627	0.00%	46.21%	0.64
Guadalajara	41	30	57476	79842	57476	0.00%	38.91%	0.54
Guadalajara	41	20	59493	79842	59493	0.00%	34.20%	0.57
Guadalajara	41	11	64981	87933	64981	0.00%	35.32%	1.25
Dublin	45	30	33548	50455	33548	0.00%	50.40%	1.77
Dublin	45	20	39393	58316	39393	0.00%	48.04%	44.73
Dublin	45	11	53742	71754	53742	0.00%	33.52%	119.70
Denver	51	30	51583	79747	51583	0.00%	54.60%	1.55
Denver	51	20	53282	83129	53282	0.00%	56.02%	19.50
Denver	51	10	67025	104823	67025	0.00%	56.39%	122.36
RioDeJaneiro	55	30	122499	168689	122499	0.00%	37.71%	130.51
RioDeJaneiro	55	20	154663	202036	154663	0.00%	30.63%	107.77
RioDeJaneiro	55	10	252758	309620	252758	0.00%	22.50%	131.54
Boston	59	30	65522	98238	65522	0.00%	49.93%	462.03
Boston	59	20	71487	120568	71487	0.00%	68.66%	182.16
Boston	59	16	74242	127186	74242	0.00%	71.31%	545.76

Table 6: Results for the real world data, preemptive SBRP, $\alpha = 3$

City	|V|	Q	Best known solution value	Heuristic solution value	Best known lower bound	Optimality Gap	Heuristic Gap	CPU time
Bari	13	30	20600	27900	20600	0.00%	35.44%	0.03
Bari	13	20	26300	37700	26300	0.00%	43.35%	0.06
Bari	13	10	44900	53900	44900	0.00%	20.04%	0.05
ReggioEmilia	14	30	32500	43500	32500	0.00%	33.85%	0.05
ReggioEmilia	14	20	47800	65400	47800	0.00%	36.82%	0.08
ReggioEmilia	14	10	83000	100800	83000	0.00%	21.45%	0.06
Bergamo	15	30	14500	21300	14500	0.00%	46.90%	0.06
Bergamo	15	20	16600	21100	16600	0.00%	27.11%	0.03
Bergamo	15	12	23100	28800	23100	0.00%	24.68%	0.08
Parma	15	30	32500	40500	32500	0.00%	24.62%	0.05
Parma	15	20	37100	49200	37100	0.00%	32.61%	0.03
Parma	15	10	51100	72800	51100	0.00%	42.47%	0.06
Treviso	18	30	31443	42001	31443	0.00%	33.58%	0.06
Treviso	18	20	33718	42781	33718	0.00%	26.88%	0.14
Treviso	18	10	46523	67610	46523	0.00%	45.33%	0.20
LaSpezia	20	30	22811	41250	22811	0.00%	80.83%	0.05
LaSpezia	20	20	27320	42830	27320	0.00%	56.77%	0.09
LaSpezia	20	10	40370	56266	40370	0.00%	39.38%	0.14
BuenosAires	21	30	158227	191669	158227	0.00%	21.14%	0.70
BuenosAires	21	20	237143	285524	237143	0.00%	20.40%	1.20
Ottawa	21	30	17370	23609	17370	0.00%	35.92%	0.06
Ottawa	21	20	20846	28856	20846	0.00%	38.42%	0.17
Ottawa	21	10	29279	41613	29279	0.00%	42.13%	0.23
SanAntonio	23	30	39553	53231	39553	0.00%	34.58%	0.17
SanAntonio	23	20	55256	71466	55256	0.00%	29.34%	0.64
SanAntonio	23	10	103199	122604	103199	0.00%	18.80%	0.41
Brescia	27	30	37000	57800	37000	0.00%	56.22%	0.89
Brescia	27	20	45000	62700	45000	0.00%	39.33%	0.78
Brescia	27	11	68600	90600	68600	0.00%	32.07%	0.23
Roma	28	30	91700	119100	91700	0.00%	29.88%	1.73
Roma	28	20	144100	192200	144100	0.00%	33.38%	1.33
Roma	28	18	148300	192100	148300	0.00%	29.53%	0.44
Madison	28	30	33627	49165	33627	0.00%	46.21%	0.47
Madison	28	20	38773	56031	38773	0.00%	44.51%	0.80
Madison	28	10	57151	77021	57151	0.00%	34.77%	0.48
Guadalajara	41	30	68778	86744	68778	0.00%	26.12%	1.83
Guadalajara	41	20	79635	103363	79635	0.00%	29.80%	26.24
Guadalajara	41	11	113864	134941	113864	0.00%	18.51%	116.20
Dublin	45	30	58076	77518	58076	0.00%	33.48%	62.34
Dublin	45	20	79200	102692	79200	0.00%	29.66%	1910.03
Dublin	45	11	130631	155798	130631	0.00%	19.27%	1083.98
Denver	51	30	67025	104823	67025	0.00%	56.39%	101.72
Denver	51	20	78593	113906	78593	0.00%	44.93%	129.84
Denver	51	10	124793	161226	124793	0.00%	29.19%	98.05
RioDeJaneiro	55	30	252758	309620	252758	0.00%	22.50%	100.60
RioDeJaneiro	55	20	366036	430518	356429.5	2.70%	20.79%	7200.01
RioDeJaneiro	55	10	665907	748684	665907	0.00%	12.43%	230.35
Boston	59	30	90848	141639	90848	0.00%	55.91%	525.94
Boston	59	20	128795	196784	128795	0.00%	52.79%	115.75
Boston	59	16	148346	205913	148346	0.00%	38.81%	6779.85

Table 7: Results for the real world data, nonpreemptive SBRP, $\alpha = 1$

City	|V|	Q	Best known solution value	Heuristic solution value	Best known lower bound	Optimality Gap	Heuristic Gap	CPU time
Bari	13	30	14600	19000	14600	0.00%	30.14%	0.02
Bari	13	20	15700	19000	15700	0.00%	21.02%	0.02
Bari	13	10	20600	27900	20600	0.00%	35.44%	0.03
ReggioEmilia	14	30	17700	27600	17700	0.00%	55.93%	0.27
ReggioEmilia	14	20	23300	33700	23300	0.00%	44.64%	0.09
ReggioEmilia	14	10	32500	43500	32500	0.00%	33.85%	0.03
Bergamo	15	30	12700	18600	12700	0.00%	46.46%	0.05
Bergamo	15	20	12700	18600	12700	0.00%	46.46%	0.03
Bergamo	15	12	13500	18000	13500	0.00%	33.33%	0.11
Parma	15	30	29000	36300	29000	0.00%	25.17%	0.01
Parma	15	20	29000	36300	29000	0.00%	25.17%	0.01
Parma	15	10	32500	40500	32500	0.00%	24.62%	0.06
Treviso	18	30	29261	42001	29261	0.00%	43.54%	0.08
Treviso	18	20	29261	42001	29261	0.00%	43.54%	0.08
Treviso	18	10	31443	42001	31443	0.00%	33.58%	0.06
LaSpezia	20	30	20746	36096	20746	0.00%	73.99%	0.03
LaSpezia	20	20	20746	36096	20746	0.00%	73.99%	0.03
LaSpezia	20	10	22811	41250	22811	0.00%	80.83%	0.06
BuenosAires	21	30	73558	107385	73558	0.00%	45.99%	1700.73
BuenosAires	21	20	87173	108596	85800.95	1.60%	26.57%	7200.00
Ottawa	21	30	16204	23609	16204	0.00%	45.70%	0.19
Ottawa	21	20	16204	23609	16204	0.00%	45.70%	0.14
Ottawa	21	10	17372	23609	17372	0.00%	35.90%	0.20
SanAntonio	23	30	22982	36644	22982	0.00%	59.45%	0.03
SanAntonio	23	20	23887	36692	23887	0.00%	53.61%	0.08
SanAntonio	23	10	39553	53231	39553	0.00%	34.58%	0.19
Brescia	27	30	30300	44200	30300	0.00%	45.87%	0.19
Brescia	27	20	31100	47800	31100	0.00%	53.70%	0.16
Brescia	27	11	34900	56900	34900	0.00%	63.04%	0.53
Roma	28	30	61900	99600	61900	0.00%	60.90%	2.28
Roma	28	20	65600	71500	65600	0.00%	8.99%	3.85
Roma	28	18	68000	108600	68000	0.00%	59.71%	0.97
Madison	28	30	29246	46816	29246	0.00%	60.08%	0.05
Madison	28	20	29839	40463	29839	0.00%	35.60%	0.11
Madison	28	10	33627	49165	33627	0.00%	46.21%	0.47
Guadalajara	41	30	57476	79842	57476	0.00%	38.91%	0.58
Guadalajara	41	20	59493	79842	59493	0.00%	34.20%	0.83
Guadalajara	41	11	64981	87933	64981	0.00%	35.32%	2.13
Dublin	45	30	33548	50455	33548	0.00%	50.40%	3.98
Dublin	45	20	39393	58316	39393	0.00%	48.04%	39.50
Dublin	45	11	53742	71754	53742	0.00%	33.52%	269.07
Denver	51	30	51583	79747	51583	0.00%	54.60%	1.68
Denver	51	20	53369	83129	53369	0.00%	55.76%	197.40
Denver	51	10	67025	104823	67025	0.00%	56.39%	98.60
RioDeJaneiro	55	30	122499	168689	122499	0.00%	37.71%	162.81
RioDeJaneiro	55	20	154663	202036	154663	0.00%	30.63%	115.28
RioDeJaneiro	55	10	252758	309620	252758	0.00%	22.50%	114.19
Boston	59	30	76376	98238	65433.33	16.72%	50.13%	7200.00
Boston	59	20	71648	120568	71648	0.00%	68.28%	525.52
Boston	59	16	74242	127186	74242	0.00%	71.31%	509.42

Table 8: Results for the real world data, nonpreemptive SBRP, $\alpha = 3$

City	|V|	Q	Best known solution value	Heuristic solution value	Best known lower bound	Optimality Gap	Heuristic Gap	CPU time
Bari	13	30	20600	27900	20600	0.00%	35.44%	0.03
Bari	13	20	26300	37700	26300	0.00%	43.35%	0.06
Bari	13	10	44900	53900	44900	0.00%	20.04%	0.04
ReggioEmilia	14	30	32500	43500	32500	0.00%	33.85%	0.03
ReggioEmilia	14	20	47800	65400	47800	0.00%	36.82%	0.12
ReggioEmilia	14	10	83000	100800	83000	0.00%	21.45%	0.12
Bergamo	15	30	14500	21300	14500	0.00%	46.90%	0.06
Bergamo	15	20	16600	21100	16600	0.00%	27.11%	0.04
Bergamo	15	12	23100	28800	23100	0.00%	24.68%	0.08
Parma	15	30	32500	40500	32500	0.00%	24.62%	0.06
Parma	15	20	37100	49200	37100	0.00%	32.61%	0.03
Parma	15	10	51100	72800	51100	0.00%	42.47%	0.05
Treviso	18	30	31443	42001	31443	0.00%	33.58%	0.06
Treviso	18	20	33718	42781	33718	0.00%	26.88%	0.16
Treviso	18	10	46523	67610	46523	0.00%	45.33%	0.20
LaSpezia	20	30	22811	41250	22811	0.00%	80.83%	0.06
LaSpezia	20	20	27320	42830	27320	0.00%	56.77%	0.10
LaSpezia	20	10	40370	56266	40370	0.00%	39.38%	0.14
BuenosAires	21	30	158228	191669	158228	0.00%	21.13%	7.71
BuenosAires	21	20	237143	285524	237143	0.00%	20.40%	328.51
Ottawa	21	30	17372	23609	17372	0.00%	35.90%	0.22
Ottawa	21	20	20847	28856	20847	0.00%	38.42%	0.44
Ottawa	21	10	29280	41613	29280	0.00%	42.12%	1.22
SanAntonio	23	30	39553	53231	39553	0.00%	34.58%	0.19
SanAntonio	23	20	55256	71466	55256	0.00%	29.34%	0.67
SanAntonio	23	10	103199	122604	103199	0.00%	18.80%	0.43
Brescia	27	30	37000	57800	37000	0.00%	56.22%	1.44
Brescia	27	20	45000	62700	45000	0.00%	39.33%	0.79
Brescia	27	11	68600	90600	68600	0.00%	32.07%	0.24
Roma	28	30	91700	119100	91700	0.00%	29.88%	13.67
Roma	28	20	144100	192200	144100	0.00%	33.38%	1.28
Roma	28	18	148300	192100	148300	0.00%	29.53%	0.44
Madison	28	30	33627	49165	33627	0.00%	46.21%	0.47
Madison	28	20	38773	56031	38773	0.00%	44.51%	0.80
Madison	28	10	57151	77021	57151	0.00%	34.77%	0.50
Guadalajara	41	30	68778	86744	68778	0.00%	26.12%	1.79
Guadalajara	41	20	79635	103363	79635	0.00%	29.80%	28.24
Guadalajara	41	11	113864	134941	113864	0.00%	18.51%	75.50
Dublin	45	30	58076	77518	58076	0.00%	33.48%	89.90
Dublin	45	20	79200	102692	79200	0.00%	29.66%	2373.68
Dublin	45	11	132254	155798	130631	1.24%	19.27%	7200.00
Denver	51	30	67025	104823	67025	0.00%	56.39%	97.34
Denver	51	20	78593	113906	78593	0.00%	44.93%	144.76
Denver	51	10	124793	161226	124793	0.00%	29.19%	35.59
RioDeJaneiro	55	30	252758	309620	252758	0.00%	22.50%	116.79
RioDeJaneiro	55	20	430518	430518	352364.8	22.18%	22.18%	7200.00
RioDeJaneiro	55	10	680920	748684	665906.2	2.25%	12.43%	7200.00
Boston	59	30	90849	141639	90849	0.00%	55.91%	1478.29
Boston	59	20	128795	196784	128795	0.00%	52.79%	197.57
Boston	59	16	150136	205913	148167.8	1.33%	38.97%	7200.00

Acknowledgments: This study was partially supported by Centre for Operational Research, Management Science and Information Systems (CORMSIS) based within the University of Southampton. This support is gratefully acknowledged.

Bibliography

N.A.H. Agatz, A.L. Erera, M.W.P. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, 45:1450–1464, 2011.

M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self-service bike hire system. *RAIRO-Operations Research*, 45:37–61, 2011.

C. Buchanan. Traffic in towns: a study of the long term problems of traffic in urban areas. Steering Group and Working Group Appointed by the Minister of Transport. London: HMSO, 1963.

D. Chemla, F. Meunier, T. Pradeau, R. Wolfler Calvo, and H. Yahiaoui. Self-service bike sharing systems: simulation, repositioning, pricing. Technical Report hal-00824078, Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique - CERMICS , Laboratoire d'Informatique de Paris-Nord - LIPN , Parallélisme, Réseaux, Systèmes d'information, Modélisation - PRISM, 2013a.

D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10:120–146, 2013b.

L. Chen, M. Hoàng Hà, A. Langevin, and M. Gendreau. Optimizing road network daily maintenance operations with stochastic service and travel times. *Transportation Research Part E: Logistics and Transportation Review*, 64:88 – 102, 2014.

J.Y.J. Chow and H.R. Sayarshad. Symbiotic network design strategies in the presence of coexisting transportation networks. *Transportation Research Part B: Methodological*, 62:13–34, 2014.

G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

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

G. Dantzig and B.C. Eaves. Fourier-Motzkin elimination and its dual. *Journal of Combinatorial Theory, Series A*, 14:288 – 297, 1973.

P. De Maio. Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, 14:41–56, 2009.

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

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

L. Gaspero, A. Rendl, and T. Urli. A hybrid ACO+CP for balancing bicycle sharing systems. In M. Blesa, C. Blum, P. Festa, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 7919 of *Lecture Notes in Computer Science*, pages 198–212. Springer Berlin Heidelberg, 2013.

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

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

M. Kaspi, T. Raviv, and M. Tzur. Parking reservation policies in one-way vehicle sharing systems. *Transportation Research Part B: Methodological*, 62:35–50, 2014.

Z. Liu, X. Jia, and W. Cheng. Solving the last mile problem: Ensure the success of public bicycle system in Beijing. *Procedia - Social and Behavioral Sciences*, 43:73 – 78, 2012. 8th International Conference on Traffic and Transportation Studies (ICTTS 2012).

R. Nair and E. Miller-Hooks. Fleet management for vehicle sharing operations. *Transportation Science*, 45:524–540, 2011.

P. Papazek, G. Raidl, M. Rainer-Harbach, and B. Hu. A pilot/vnd/grasp hybrid for the static balancing of public bicycle sharing systems. In R. Moreno-Dìaz, F. Pichler, and A. Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2013*, volume 8111 of *Lecture Notes in Computer Science*, pages 372–379. Springer Berlin Heidelberg, 2013.

M. Rainer-Harbach, P. Papazek, G.R. Raidl, B. Hu, and C. Kloimllner. PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, pages 1–33, 2014.

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

J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve. Inventory rebalancing and vehicle routing in bike sharing systems. Working paper, 2 2013. Carnegie Mellon University, Tepper School of Business.

M.S. Smith and G. Kauermann. Bicycle commuting in Melbourne during the 2000s energy crisis: A semiparametric analysis of intraday volumes. *Transportation Research Part B: Methodological*, 45: 1846–1862, 2011.

W.T. Tutte. *Graph Theory*. Cambridge Mathematical Library. Cambridge University Press, 2001.