CIRRELT

Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation

Balancing a Dynamic Public Bike-Sharing System

Claudio Contardo
Catherine Morency
Louis-Martin Rousseau

March 2012

CIRRELT-2012-09

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Balancing a Dynamic Public Bike-Sharing System

Claudio Contardo[1,*], Catherine Morency[2,3], Louis-Martin Rousseau[1,2]

[1] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Department of Civil, Geological and Mining Engineering, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

Abstract. In this article we introduce a dynamic public bike-sharing balancing problem (DPBSBP) arising from the daily operations of a public bike-sharing (PBS) system. In a PBS system, especially during peak hours, some stations have more demand than others. If no action is taken by the service provider they rapidly fill or empty, thus preventing other users from collecting or delivering bikes. The service provider must route vehicles to transport bikes from full stations to stations with shortages to balance the network. We formally define the problem and present a mathematical formulation. This formulation, however, cannot handle medium or large instances. We therefore present an alternative modeling approach that takes advantage of two decomposition schemes, Dantzig-Wolfe decomposition and Benders decomposition, to derive lower bounds and feasible solutions in short computing times.

Keywords. Public bike-sharing system, column generation, Benders decomposition.

Acknowledgements. Thanks are due to the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support. We would also like to thank Roberto Wolfler-Calvo for sending us a copy of their work in the static problem.

---

* Corresponding author: Claudio.Contardo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1. Introduction

In recent years, public bike-sharing (PBS) systems have been gaining increasing popularity in transportation plans as a strategy to multiply travel choices, promote the use of active modes of transport, decrease the dependence on the automobile, and especially reduce greenhouse gas emissions. PBS systems are currently spreading across the globe: in 2009, Shaheen et al. (2010) estimated that about 100 bikesharing programs were implemented in 125 cities, for a total of more than 140,000 bikes. In 2010, 45 new operations were planned in 22 countries. There is also increasing interest in the research community in understanding how PBS systems are used and what factors affect travel behavior.

In Montreal, the BIXI system was launched in the summer of 2009 and rapidly gained popular support. In 2011, more than 5,000 bikes were available to users across 405 stations and more than 4 million trips were performed. The increasing service area combined with the high number of users makes it difficult to meet the demand. Also, as are many other areas, the Montreal region is quite monocentric, so the daily commutes are mainly unidirectional. This results in increased travel demand toward the CBD (central business district) during the AM peak period and vice versa in the PM peak. For the PBS system, this means that in the morning there is high demand for bikes in peripheral areas and especially high demand for delivery points in the CBD. Particularly in the peak periods, the number of outgoing and incoming bikes at certain stations is unbalanced, creating the need for intervention by the operator. Sometimes the topography enhances this effect, with hilltop stations being mostly starting points while stations below hills are mostly destinations. At certain times of the day a subset of stations in the network will have extremely high demand, and action becomes necessary. These stations will be primarily pickup points or primarily delivery points. In the former case, if no action is taken by the service provider, the station will rapidly empty, thus preventing other users from collecting bikes. In the latter case, the station will rapidly fill, thus preventing other users from delivering bikes. When the network is not able to meet the demand with a reasonable standard of quality, we will say that it is *unbalanced*. *Balancing* the network refers to the actions taken by the service provider with the objective of ensuring a certain quality of service.

Given a set of stations, a limited fleet of vehicles, and time-dependent demands for bikes, the problem is to schedule vehicle routes to visit some of the stations to perform pickup and delivery so as to minimize the number of users who cannot be served, i.e., the number of users who try to collect bikes from empty stations or to deliver bikes to full stations.

Research into PBS systems is relatively recent. Wang et al. (2010) review PBS systems by analyzing the operation of several different systems. From an OR perspective, the literature has focused mainly on the strategic planning of the network design. dell'Olio et al. (2011) present a complete methodology for the design of such a network based on demand estimates. Their methodology considers the locations of the stations and the fares. Lin and Yang (2011) address the strategic problem of finding optimal stations using mathematical programming techniques. They formulate the problem as a nonlinear mixed-integer problem and solve it with a commercial solver. Vogel and Mattfeld (2011) present a methodology for strategic and operational planning using data mining. Customer demand is forecast using data mining techniques, and the station locations are set accordingly. A similar methodology is applied to schedule vehicle routes to balance the network.

Regarding operational planning, besides the methodology proposed by (Vogel and Mattfeld 2011), we can mention the recent works of Raviv et al. (2011) and Chemla et al. (2011). In Raviv et al. (2011), the authors propose several mathematical formulations for the static balancing problem. In the static case, customer demand is assumed to be negligible (for example, during the night). The objective is to schedule vehicle routes to visit the stations in the minimum possible time so as to accomplish a certain target (typically a desired number of bikes present at each station) by the end of the time period. In Chemla et al. (2011) the authors also address the static problem and propose an exact algorithm based on column generation. A column in the proposed formulation represents a feasible vehicle route along with a sequence of pickup/delivery actions. A suitable pricing algorithm is proposed based on dynamic programming.

The problem studied in this paper is also closely related to other well-studied vehicle-routing problems. The one-commodity pickup and delivery problem (1-PDP) deals with the problem of moving a single commodity through a number of customer locations. A fleet of vehicles visits each customer location at most once and transports a given number of units of the commodity from the pickup nodes to the delivery nodes. Applications of this problem include money transportation between different branches of a bank or grocery distribution between supermarkets and suppliers. Algorithms include branch-and-cut methods (Hernández-Pérez and Salazar-González 2003, 2007), approximation algorithms (Anily and Bramel 1999), and heuristics (Hernández-Pérez et al. 2009, Zhao et al. 2009). The main difference between the 1-PDP and the problem addressed in this paper is that in the latter the number of bikes to transport from one station to another is a decision variable. The swapping problem (SP) is the problem of moving multiple commodities between nodes. With each node we associate a pair of indices (a_i, b_i) representing the types of commodity for which the customer is a supplier and a demand point, respectively. A single vehicle must visit the customer locations at minimum traveling cost so as to fulfill each node's demand. The SP was first introduced by Anily and Hassin (1992), who also introduced polynomial-time approximation algorithms for the problem. Bordenave et al. (2009, 2010) investigated different variations of the SP and proposed heuristics and exact methods based on branch-and-cut techniques. Erdogan et al.

(2010) developed a branch-and-cut algorithm for which they adapted several valid inequalities from the 1-PDP.

In this paper we consider a dynamic balancing problem, which comes from balancing the PBS network during peak hours. In contrast to the static case, demand cannot be neglected. The problem is first formulated using an arc-flow formulation on a suitable space-time network. Dantzig-Wolfe decomposition (Dantzig and Wolfe 1960) is then applied and two different formulations are derived. One is solved using column generation, and the other is used as a primal heuristic to find good solutions quickly. The main contributions of this paper are the following:

1. We introduce a dynamic public bike-sharing balancing problem (DPBSBP) arising from the daily operations of a PBS system during peak hours.
2. We provide mathematical formulations for the DPBSBP.
3. We develop a scalable methodology that provides lower and upper bounds in short computing times.

The remainder of this paper is as follows: In Section 2 we formally define the DPBSBP and present a mathematical formulation based on vehicle and commodity flows. In Section 3 we present a new methodology to solve the DPBSBP that relies on two decomposition schemes, namely Danzig-Wolfe decomposition (Dantzig and Wolfe 1960) and Benders decomposition (Benders 1962). In Section 4 we present computational results. Finally, Section 5 provides concluding remarks.

2. Problem definition and mathematical formulation

Let $\mathcal{K}$ be the set of vehicles. With each vehicle $k \in \mathcal{K}$ we associate a capacity, Q_k, an initial load at the beginning of the time horizon, Q_k^0, and an initial position given by the point u^k. Let $\mathcal{V}$ be the set of stations in the network. With each station $v \in \mathcal{V}$ we associate a capacity, C_v, and an initial number of bikes at the beginning of the time horizon, C_v^0. The time horizon is discretized into a set of periods $\mathcal{T}$. This is done to explicitly take into account the possibility of visiting the same station at different times. For the sake of clarity, we assume that the periods are indexed from 1 to $|\mathcal{T}|$. We consider a set of *states*, denoted $\mathcal{S}$, composed of: 1) the initial positions of the vehicles at time 0, $\{(u^k, 0) : k \in \mathcal{K}\}$, 2) nodes for the stations at the different time periods, $\{(v,t), v \in \mathcal{V}, t \in \mathcal{T}\}$, and 3) a dummy node denoted ϕ to represent the end of a route in the planned schedule. We denote by $v(s)$ and $t(s)$ the node and period corresponding to state $s \neq \phi$. We denote by $\mathcal{S}_{\mathcal{V}}$ the subset of states composed of the pairs $(v,t), v \in \mathcal{V}, t \in \mathcal{T}$. For a given state $s \in \mathcal{S}_{\mathcal{V}}$ we set $pred(s) = (v(s), t(s) - 1)$ if $t(s) \geq 2$ and $succ(s) = (v(s), t(s) + 1)$ if $t(s) \leq |\mathcal{T}| - 1$. Also, with each state $s \in \mathcal{S}_{\mathcal{V}}$ we associate a demand f_s for bikes ($f_s \geq 0$ if s is a pickup point, and $f_s < 0$ if s is a delivery point). Let us consider a graph $G = (\mathcal{S}, \mathcal{A})$, where the arc set $\mathcal{A}$ is defined as follows. Suppose that the travel times are scaled to have the same units as the time discretization (for instance, if a time period represents a window of 5 min, a trip of 10 min starting at period t will end at period $t+2$). The arc set $\mathcal{A}$ is composed of three types of arcs. First, it contains all feasible direct trips between a pair of states, i.e., all arcs $(s,s') \in \mathcal{S} \times \mathcal{S}$ such that $t(s') - t(s) \geq d(v(s), v(s')) > t(s') - t(s) - 1$, where $d(\cdot,\cdot)$ is the distance between two nodes. Second, it contains all arcs $(s, succ(s))$ for $s \in \mathcal{S}_{\mathcal{V}}$ such that $t(s) \leq |\mathcal{T}| - 1$. These arcs represent the action of waiting at station $v(s)$ for a period of time. Finally, it contains all arcs (s, ϕ) representing the end of the vehicle routes. Note that we can consider the distance function $d(\cdot,\cdot)$ to be time-dependent, which gives extra flexibility to the modeling approach. However, we assume that the travel times do not depend on the actions performed by a vehicle in a given station. This is a limitation of the model that can be partially solved if the travel times take into account the average service time at stations.

In Fig. 1 we illustrate a network with two vehicles and three stations, and the space-time network resulting after a time discretization into five periods of one unit each. As can be seen in this small example, the number of edges in the space-time network is much higher than the number in the original graph. Also, note that the space-time network allows only arcs that go forward in time. This property will be exploited later to derive a polynomial-time algorithm for the pricing problem in the branch-and-price solver. Finally, note that although the distance from node u_2^0 to node 1 in the original network is 3/2, it is rounded up to 2 in the space-time network.

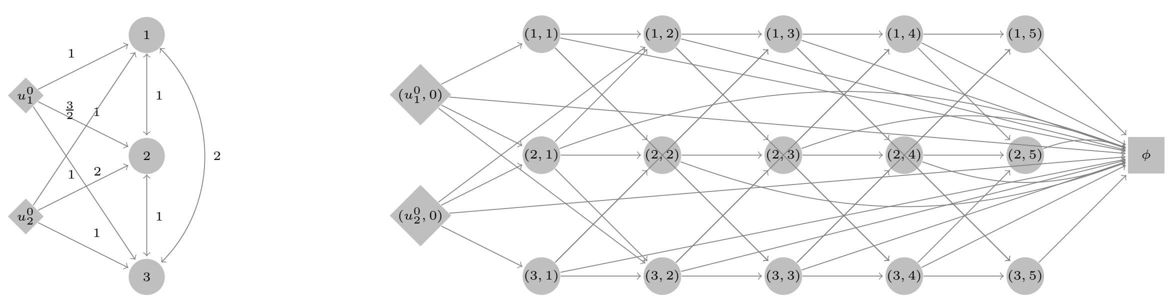

Figure 1 Original network vs. space-time network

2.1. Arc-flow formulation

Let us consider the space-time network previously defined. For each state $s \in \mathcal{S}_{\mathcal{V}}$ let $y_s^+, y_s^- \geq 0$ be two continuous variables representing a shortage and excess of bikes at state s, respectively (together, these two quantities represent the unmet demand). Also, let $z_s \geq 0$ be a continuous variable representing the number of bikes left at state s. For each arc $a \in \mathcal{A}$ and vehicle $k \in \mathcal{K}$ let w_a^k be a binary variable equal to 1 iff vehicle k traverses arc a in its route, and let $x_a^k \geq 0$ be a continuous variable equal to the load of vehicle k along arc a. These loads are integer but they can be relaxed to be continuous because they can be retrieved by solving a minimum-cost flow problem on the space-time network for fixed integer values of the vehicle-flow variables w. For state $s \in \mathcal{S}$

we denote by $\delta^+(s)$ the set of arcs ending at s and by $\delta^-(s)$ the set of arcs starting at s. A valid formulation of the problem is as follows:

$$\min \sum_{s\in\mathcal{S}_\mathcal{V}} \left(y_s^+ + y_s^-\right) \tag{1}$$

s. t.

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x_a^k - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x_a^k - z_s + y_s^+ - y_s^- = f_s - C^0_{v(s)} \qquad s\in\mathcal{S}_\mathcal{V}, t(s)=1 \tag{2}$$

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x_a^k - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x_a^k + z_{pred(s)} - z_s + y_s^+ - y_s^- = f_s \qquad s\in\mathcal{S}_\mathcal{V}, t(s)\geq 2 \tag{3}$$

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} w_a^k \leq 1 \qquad s\in\mathcal{S}_\mathcal{V} \tag{4}$$

$$y_s^+, y_s^- \geq 0 \qquad s\in\mathcal{S}_\mathcal{V} \tag{5}$$

$$0 \leq z_s \leq C_{v(s)} \qquad s\in\mathcal{S}_\mathcal{V} \tag{6}$$

$$x_a^k \leq Q_k w_a^k \qquad k\in\mathcal{K}, a\in\mathcal{A} \tag{7}$$

$$\sum_{a\in\delta^-(s)} w_a^k - \sum_{a\in\delta^+(s)} w_a^k = 0 \qquad k\in\mathcal{K}, s\in\mathcal{S}_\mathcal{V} \tag{8}$$

$$\sum_{a\in\delta^-(u^k,0)} w_a^k = 1 \qquad k\in\mathcal{K} \tag{9}$$

$$\sum_{a\in\delta^-(u^k,0)} x_a^k = Q_k^0 \qquad k\in\mathcal{K} \tag{10}$$

$$x \geq 0 \tag{11}$$

$$w \text{ binary.} \tag{12}$$

The objective function represents the total unmet demand, i.e., the number of users who tried to collect bikes from empty stations or to deliver bikes to full stations. Constraints (2)–(3) are the flow conservation constraints at each station for every time period. The role of the variables y^+, y^- is to compensate for the *imbalance* of the network. In a perfectly balanced network these quantities will always be zero. Constraints (4) ensure that each node is visited at most once in a time period. Constraints (5) are the non-negativity constraints for the variables y^+, y^-. Constraints (6) are the non-negativity constraints for the variables z and the capacity constraints of the stations in every time period. Constraints (7) link the use of each arc to the maximum allowable load on the vehicle traversing that arc. Constraints (8) are the vehicle-flow conservation constraints; they force vehicles to leave the stations previously visited. Constraints (9) ensure that every vehicle is used exactly once. Note that this includes the option of not using a vehicle k by introducing the arc $((u^k, 0); \phi)$. Constraints (10) ensure that vehicles leave their starting positions with their current loads. Constraints (11) are the non-negativity constraints for the vehicle loads on arcs. Finally, constraints (12) ensure that the vehicle-flow variables w are binary.

Note that a feasible solution of the above formulation may not be entirely feasible for our balancing problem. Indeed, although the penalties tend to minimize the use of the variables y, these variables may take unrealistic values, creating or destroying bikes, if this pays off in the future. Formally, let $(\overline{x}, \overline{y}, \overline{z}, \overline{w})$ be a solution of the above integer program. Let $s \in \mathcal{S}_{\mathcal{V}}$ be a station at a given time period (for the sake of brevity we assume that $t(s) \geq 2$), and let $q(s) = \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{x}_a^k + \overline{z}_{pred(s)} - f(s)$. The three situations below are considered *pathological*:

1. $0 < q(s) < C_{v(s)} + \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{w}_a^k Q_k$ and $\overline{y}_s^+ + \overline{y}_s^- > 0$.
2. $q(s) \leq 0$ and $\overline{y}_s^+ > -q(s)$.
3. $q(s) \geq C_{v(s)} + \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{w}_a^k Q_k$ and $\overline{y}_s^- > q(s) - C_{v(s)} - \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{w}_a^k Q_k$.

In case (1), $q(s)$ represents the number of bikes that are available at node s at the end of the period. These bikes should be available for the future, either at the same station via variable $\overline{z}_s$ or transported on a vehicle via variables $\overline{x}_a^k, a \in \delta^-(s), k \in \mathcal{K}$. However, if $\overline{y}_s^+ > 0$ then bikes have been created and inserted into the system, and if $\overline{y}_s^- > 0$ then bikes have been destroyed and removed from the system. In case (2), there is a shortage of $-q(s)$ bikes at station s at the end of the time period, but if $\overline{y}_s^+ > -q(s)$ then bikes are created and inserted into the system. In case (3), there is a shortage of $q(s) - C_{v(s)} - \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{w}_a^k Q_k$ docking points at the station. If $\overline{y}_s^- > q(s) - C_{v(s)} - \sum_{a \in \delta^+(s)} \sum_{k \in \mathcal{K}} \overline{w}_a^k Q_k$ then bikes are destroyed. Fortunately, these three pathological cases can be removed from the solution to give another solution with at most the same cost and with no pathological situations, as stated in the following proposition:

Proposition 1. *Let $(\overline{x}, \overline{y}, \overline{z}, \overline{w})$ be a solution of program* (1)–(12)*, and let $s \in \mathcal{S}_{\mathcal{V}}$ be such that a pathological situation occurs. It is possible to build another solution $(\overline{x}', \overline{y}', \overline{z}', \overline{w}')$ of problem* (1)–(12) *with at most the same cost where this situation does not occur.*

Proof See the Appendix.

Remark 1. The proof of this proposition uses the special structure of the cost function (it depends only on the y variables and the coefficients are all equal). For different cost structures the proposition may no longer be valid, and so additional variables and/or constraints might be necessary to ensure feasibility. We leave the study of this property under different cost structures for future research.

3. Column generation coupled with Benders decomposition (CG+BD)

In this section we introduce a heuristic procedure based on the solution of two different decompositions of the problem that are performed sequentially. First, we apply Dantzig-Wolfe decomposition to the arc-flow formulation and solve the linear relaxation of the resulting problem using column generation. The Dantzig-Wolfe reformulation of the problem contains many fewer constraints than the original arc-flow formulation, and a polynomial-time pricing algorithm allows us to quickly obtain a lower bound for the problem. Second, we apply Benders decomposition to another formulation of the problem and use the information provided by the first solution. We use the basic columns of that solution as integer variables and find the continuous variables by solving a series of minimum-cost flow problems in a Benders decomposition framework; this provides a feasible solution of the problem and therefore an upper bound.

3.1. Column generation

In this section we introduce a new formulation obtained by applying Dantzig-Wolfe decomposition (Dantzig and Wolfe 1960) to the original arc-flow formulation, and we use an efficient pricing procedure to dynamically generate columns with a negative reduced cost. In the new formulation, the columns represent (route, load) patterns, i.e., the arcs traversed by a vehicle along with the load in the vehicle at every arc in the route. The new formulation drastically reduces the number of constraints, leading to a more compact formulation (in terms of this number) than the arc-flow formulation introduced earlier. We present a polynomial-time algorithm to solve the pricing problem derived from this decomposition scheme. Our scheme produces lower bounds quickly and allows us to handle instances that are too large for the MIP solver used to solve formulation (1)–(12).

3.1.1. Pattern-based formulation The arc-flow formulation introduced in the previous section is decomposed as follows. Let us consider the polytope $\{(w, x) : (w, x) \text{ satisfy constraints (7)–(12)}\}$. It is possible to decompose this polytope into $|\mathcal{K}|$ subsets, one for each vehicle. Let $\mathcal{X}_k = \{l = (w^k, x^k) : (w^k, x^k) \text{ satisfy constraints (7)–(12)}\}$ be the set of patterns associated with vehicle k, and let $\mathcal{X} = \cup_{k\in\mathcal{K}}\mathcal{X}_k$ be the set of all patterns. For each $l \in \mathcal{X}$ and $a \in \mathcal{A}$ let w_a^l be a binary constant equal to 1 iff the route associated with pattern l uses arc a, and let x_a^l be the commodity-flow on arc a when it is traversed by the route on pattern l. Let θ_l be the variable associated with pattern l. The Dantzig-Wolfe reformulation of the problem is

$$\min \sum_{s\in\mathcal{S}_\mathcal{V}} \left(y_s^+ + y_s^-\right) \tag{13}$$

s. t.

$$\sum_{l\in\mathcal{X}} \sum_{a\in\delta^+(s)} x_a^l\theta_l - \sum_{l\in\mathcal{X}} \sum_{a\in\delta^-(s)} x_a^l\theta_l - z_s + y_s^+ - y_s^- = f_s - C_{v(s)}^0 \qquad s \in \mathcal{S}_\mathcal{V}, t(s) = 1 \tag{14}$$

$$\sum_{l\in\mathcal{X}} \sum_{a\in\delta^+(s)} x_a^l\theta_l - \sum_{l\in\mathcal{X}} \sum_{a\in\delta^-(s)} x_a^l\theta_l + z_{pred(s)} - z_s + y_s^+ - y_s^- = f_s \qquad s \in \mathcal{S}_\mathcal{V}, t(s) \geq 2 \tag{15}$$

$$\sum_{l \in \mathcal{X}} \sum_{a \in \delta^+(s)} w_a^l \theta_l \leq 1 \qquad s \in \mathcal{S}_{\mathcal{V}} \tag{16}$$

$$\sum_{l \in \mathcal{X}_k} \theta_l = 1 \qquad k \in \mathcal{K} \tag{17}$$

$$0 \leq z_s \leq C_{v(s)} \qquad s \in \mathcal{S}_{\mathcal{V}} \tag{18}$$

$$y^+, y^- \geq 0 \tag{19}$$

$$\theta \geq 0 \tag{20}$$

$$\sum_{l \in \mathcal{X}} w_a^l \theta_l \in \{0, 1\} \qquad a \in \mathcal{A}. \tag{21}$$

The meaning of each set of constraints is clear or can be derived from the previous formulation. Note that constraints (21) are required only to impose the integrality of the routing part; they can be discarded for the computation of the root-node relaxation. As a consequence, the problem has a linear number of constraints, rather than the cubic number ($|\mathcal{K}| \times |\mathcal{A}|$) of the arc-flow formulation. The problem contains an exponential number of variables θ, and even small instances cannot be directly solved by a general-purpose optimization solver. However, the variables θ can be dynamically generated and added to the problem in a column-generation fashion. We now describe the pricing algorithm used to find columns with a negative reduced cost.

3.1.2. Pricing subproblem The pricing subproblem can be decomposed into $|\mathcal{K}|$ integer problems, one for each vehicle k, as follows:

$$-\gamma_k + \min_{w,x} < \overline{c^w}, w > + < \overline{c^x}, x > \tag{22}$$

s. t.

$$x_a^k \leq Q_k w_a^k \qquad a \in \mathcal{A} \tag{23}$$

$$\sum_{a \in \delta^-(s)} w_a^k - \sum_{a \in \delta^+(s)} w_a^k = 0 \qquad s \in \mathcal{S}_{\mathcal{V}} \tag{24}$$

$$\sum_{a \in \delta^-(u^k, 0)} w_a^k = 1 \tag{25}$$

$$\sum_{a \in \delta^-(u^k, 0)} x_a^k = Q_k^0 \tag{26}$$

$$x \geq 0 \tag{27}$$

$$w \text{ binary} \tag{28}$$

where γ_k is the dual variable of constraint (17) and $\overline{c^w}, \overline{c^x}$ are the reduced costs of the variables w, x, respectively, whose expressions we will give later. Constraints (24)–(25) define a shortest-path problem structure. If a point w is feasible w.r.t. these two constraints, an extreme (optimal) solution for the commodity flow variables x is given by

$$x_a^k = \begin{cases} Q_k^0 w_a^k & \text{if } a \in \delta^-(u^k,0) \\ 0 & \text{if } a \notin \delta^-(u^k,0) \text{ and } \overline{c^x}_a > 0 \\ Q_k w_a^k & \text{if } a \notin \delta^-(u^k,0) \text{ and } \overline{c^x}_a \leq 0. \end{cases} \tag{29}$$

In other words, the subproblem can be rewritten as the following shortest path problem:

$$-\gamma_k + \min < \widehat{c^w}, w > \tag{30}$$

s. t.

$$\sum_{a\in\delta^-(s)} w_a^k - \sum_{a\in\delta^+(s)} w_a^k = 0 \qquad s \in \mathcal{S}_\mathcal{V} \tag{31}$$

$$\sum_{a\in\delta^-(u^k,0)} w_a^k = 1 \tag{32}$$

$$w \text{ binary} \tag{33}$$

with

$$\widehat{c^w}_a = \begin{cases} \overline{c^w}_a + Q_k^0 \overline{c^x}_a & \text{if } a \in \delta^-(u^k,0) \\ \overline{c^w}_a & \text{if } a \notin \delta^-(u^k,0) \text{ and } \overline{c^x}_a > 0 \\ \overline{c^w}_a + Q_k \overline{c^x}_a & \text{if } a \notin \delta^-(u^k,0) \text{ and } \overline{c^x}_a \leq 0. \end{cases} \tag{34}$$

This last problem can be solved efficiently by any label-correcting algorithm, since the space-time network does not allow cycles.

Now, let us consider a column $l \in \mathcal{X}_k$, with its routing part r_l represented by a sequence of arcs in the space-time network, i.e., $r_l = (a_l^1, \ldots, a_l^n)$. The sequence of internal stations visited on the route is denoted $(s_l^1, \ldots, s_l^{n-1})$. The load part is denoted ρ_l and represents the number of bikes being transported on each arc of the route, i.e., $\rho_l = (\rho_l^1 = Q_k^0, \ldots, \rho_l^n)$. Let α, β, and γ represent the dual variables associated with constraints (14)–(17) (indeed, we use the same dual variable α for Eqs. (14) and (15)). Then, if we define $\alpha_{s_l^0} = \alpha_{s_l^n} = \beta_{s_l^0} = \beta_{s_l^n} = 0$, the reduced cost of route l is given by

$$\overline{c}_l = -\left(\sum_{j=1}^{n-1} \alpha_{s_l^j} (\rho_l^j - \rho_l^{j+1}) - \sum_{j=1}^{n} \beta_{s_l^j} + \gamma_k \right) \tag{35}$$

$$= -\left(\sum_{j=1}^{n} \rho_l^j (\alpha_{s_l^j} - \alpha_{s_l^{j-1}}) - \sum_{j=1}^{n} \beta_{s_l^j} + \gamma_k \right). \tag{36}$$

Now, the commodity part of the pattern is always extreme in the solution of the pricing, i.e., $\rho_l^j \in \{0, Q_k\}$ for $j > 1$. Let us define $\Delta_l^j = \alpha_{s_l^j} - \alpha_{s_l^{j-1}}$ for $j = 2, \ldots, n$. By constraint (26), $\rho_l^1 = Q_k^0$. For $j = 2, \ldots, n$, if $\Delta_l^j > 0$ then $\rho_l^j = Q_k$, otherwise $\rho_l^j = 0$. Then the reduced costs $\overline{c^w}_a, \overline{c^x}_a$ for a given arc $a = (u,v)$ are as follows:

$$\overline{c^x}_a = \alpha_u - \alpha_v \tag{37}$$

$$\overline{c^w}_a = -(\beta_u + \beta_v)/2. \tag{38}$$

Given this, it is possible to build the modified reduced costs $\widehat{c^w}$ using expression (34) and to use the shortest path problem (30)–(33) as a pricing algorithm to find columns with a negative reduced cost.

Formulation (13)–(21) reduces the number of constraints in the problem by pushing many of them into the subproblem. However, our computational experience has shown that the extreme patterns produced by the subproblem solutions are unlikely to produce good feasible solutions because of the extremity of the load vector.

3.2. Benders decomposition

In this section we present a heuristic method based on Benders decomposition (Benders 1962). We introduce a new formulation of the problem with two types of variables: route variables, each of which represents the full route of a vehicle, and commodity variables that represent the loads of the vehicles along their routes. To use this formulation efficiently, we do not perform column generation on the set of route variables. We instead consider a fixed subset of route variables coming from the basic patterns produced by the solution of the linear relaxation of problem (13)–(21). With this fixed and usually small set of routes, we solve the new formulation using Benders decomposition.

3.2.1. Hybrid route-flow formulation Let us consider the original arc-flow formulation and the polytope $\{w : w \text{ satisfies constraints } (8)\text{–}(9), (12)\}$ containing the feasible solutions of the routing part of the problem. As before, this polytope can be decomposed into $|\mathcal{K}|$ sets, one per vehicle, which we call $\mathcal{X}'_k$. Let $\mathcal{X}' = \cup_{k\in\mathcal{K}}\mathcal{X}'_k$ be the set of all possible routes. For route $l \in \mathcal{X}'$ we define w^l_a to be a binary constant equal to 1 iff route l uses arc a. We retain the arc-flow variables x^k_a to represent the flow of vehicles on the arcs. The formulation associated with this decomposition is as follows:

$$\min \sum_{s\in\mathcal{S}} \left(y^+_s + y^-_s\right) \tag{39}$$

s. t.

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x^k_a - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x^k_a - z_s + y^+_s - y^-_s = f_s - C^0_{v(s)} \qquad s\in\mathcal{S}_\mathcal{V}, t(s)=1 \tag{40}$$

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x^k_a - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x^k_a + z_{pred(s)} - z_s + y^+_s - y^-_s = f_s \qquad s\in\mathcal{S}_\mathcal{V}, t(s)\geq 2 \tag{41}$$

$$\sum_{l\in\mathcal{X}'}\sum_{a\in\delta^+(s)} w^l_a\theta_l \leq 1 \qquad s\in\mathcal{S}_\mathcal{V} \tag{42}$$

$$\sum_{l\in\mathcal{X}_k} \theta_l = 1 \qquad k\in\mathcal{K} \tag{43}$$

$$x_a^k \leq Q_k \sum_{l \in \mathcal{X}'_k} w_a^l \theta_l \qquad k \in \mathcal{K}, a \in \mathcal{A} \qquad (44)$$

$$\sum_{a \in \delta^-(u^k,0)} x_a^k = Q_k^0 \qquad k \in \mathcal{K} \qquad (45)$$

$$0 \leq z_s \leq C_{v(s)} \qquad s \in \mathcal{S}_\mathcal{V} \qquad (46)$$

$$y^+, y^- \geq 0 \qquad (47)$$

$$x \geq 0 \qquad (48)$$

$$\theta_l \in \{0,1\} \qquad l \in \mathcal{X}. \qquad (49)$$

This formulation contains many more constraints than the previous one. However, if a good set of routes $\mathcal{B} \subset \mathcal{X}'$ is given in advance, a reduced problem can be solved to find good arc loads that are not necessarily extreme. This provides a much better upper bound than the previous formulation does when restricted to a subset of columns.

3.2.2. Benders master problem Let α, β, γ, and λ be the dual variables associated with constraints (40)–(41), (44), (45), and (47), respectively. Let $\mathcal{B} \subseteq \mathcal{X}'$ be the set of routes for the patterns with strictly positive values in the solution of the linear relaxation of problem (13)–(21), and let $\mathcal{B}_k$ be the set of routes associated with vehicle k. Let $\mathcal{J}$ be the set of extreme points of the dual polyhedron, and let j denote the jth extreme point. The Benders reformulation of problem (39)–(49) when restricted to the routes in $\mathcal{B}$ is

$$\min z \qquad (50)$$

s. t.

$$z \geq \sum_{s \in \mathcal{S}_\mathcal{V}} \left(\widetilde{f}_s \alpha_s^j + C_{v(s)} \lambda_s^j \right) + \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} \sum_{l \in \mathcal{B}_k} Q_k w_a^l \beta_a^{kj} \theta_l + \sum_{k \in \mathcal{K}} Q_k^0 \gamma_k^j \qquad j \in \mathcal{J} \qquad (51)$$

$$\sum_{l \in \mathcal{B}} \sum_{a \in \delta^+(s)} w_a^l \theta_l \leq 1 \qquad s \in \mathcal{S}_\mathcal{V} \qquad (52)$$

$$\sum_{l \in \mathcal{B}_k} \theta_l = 1 \qquad k \in \mathcal{K} \qquad (53)$$

$$\theta_l \in \{0,1\} \qquad l \in \mathcal{B} \qquad (54)$$

where $\widetilde{f}_s$ is equal to $f_s - C_{v(s)}^0$ if $t(s) = 1$ and to f_s if $t(s) \geq 2$. This problem contains a (typically) large number of constraints (51), one per extreme dual point, and is computationally intractable. To overcome this issue, we solve a restricted master problem subject to a small number of these constraints, those associated with a subset of extreme points $\overline{\mathcal{J}}$. A subproblem is then solved to find violated cuts associated with the relaxed constraints (51); this is explained in the next subsection.

3.2.3. Benders subproblem Let $\overline{\theta}$ be an optimal solution of the master problem associated with a subset of extreme dual points $\overline{\mathcal{J}}$. The following primal subproblem must be solved to find another dual point to include in $\overline{\mathcal{J}}$ on the next iteration:

$$\min \quad \sum_{s\in\mathcal{S}_\mathcal{V}} (y_s^+ + y_s^-) \tag{55}$$

s. t.

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x_a^k - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x_a^k - z_s + y_s^+ - y_s^- = f_s - C_{v(s)}^0 \qquad s\in\mathcal{S}_\mathcal{V}, t(s)=1 \tag{56}$$

$$\sum_{k\in\mathcal{K}}\sum_{a\in\delta^+(s)} x_a^k - \sum_{k\in\mathcal{K}}\sum_{a\in\delta^-(s)} x_a^k + z_{pred(s)} - z_s + y_s^+ - y_s^- = f_s \qquad s\in\mathcal{S}_\mathcal{V}, t(s)\geq 2 \tag{57}$$

$$x_a^k \leq Q_k \sum_{l\in\mathcal{B}_k} w_a^l \overline{\theta}_l \qquad k\in\mathcal{K}, a\in\mathcal{A} \tag{58}$$

$$\sum_{a\in\delta^-(u^k,0)} x_a^k = Q_k^0 \qquad k\in\mathcal{K} \tag{59}$$

$$0 \leq z_s \leq C_{v(s)} \qquad s\in\mathcal{S}_\mathcal{V} \tag{60}$$

$$y^+, y^- \geq 0 \tag{61}$$

$$x \geq 0. \tag{62}$$

This problem corresponds to a minimum-cost flow problem on the space-time network and can easily be solved using either the network simplex algorithm or another specialized method. For vehicle k, define $\mathcal{A}_k^* = \mathcal{A}\setminus(\delta^-(u_k^0,0)\cup\delta^+(\phi))$. We also use the convention that $\alpha_{succ(s)} = 0$ if $t(s) = |\mathcal{T}|$. The associated dual problem is the following:

$$\max \sum_{s\in\mathcal{S}} (\widetilde{f}_s\alpha_s + C_{v(s)}\lambda_s) + \sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}}\sum_{l\in\mathcal{B}_k} Q_k w_a^l \overline{\theta}_l \beta_a^k + \sum_{k\in\mathcal{K}} Q_k^0 \gamma_k \tag{63}$$

s. t.

$$\alpha_{tail(a)} + \beta_a^k + \gamma_k \leq 0 \qquad k\in\mathcal{K}, a\in\delta^-(u^k,0) \tag{64}$$

$$-\alpha_{head(a)} + \beta_a^k \leq 0 \qquad k\in\mathcal{K}, a\in\delta^+(\phi) \tag{65}$$

$$\alpha_{tail(a)} - \alpha_{head(a)} + \beta_a^k \leq 0 \qquad k\in\mathcal{K}, a\in\mathcal{A}_k^* \tag{66}$$

$$\alpha_{succ(s)} - \alpha_s + \lambda_s \leq 0 \qquad s\in\mathcal{S}_\mathcal{V} \tag{67}$$

$$-1 \leq \alpha_s \leq 1 \qquad s\in\mathcal{S}_\mathcal{V} \tag{68}$$

$$\beta, \lambda \leq 0 \tag{69}$$

$$\gamma \text{ unrestricted.} \tag{70}$$

Although both problems involve a cubic number of variables and constraints, we can still rely on specialized algorithms for the former. Indeed, for a feasible route vector $\overline{\theta}$, many variables and constraints can be dropped from problem (55)–(62). Specifically, the arc-flow variables x_a^k and their related constraints (58) can be dropped from the problem whenever the right-hand side of constraint (58) is equal to zero. Solving this reduced problem is much faster than solving the original problem. However, we lose the dual information related to the constraints that were dropped. To recover an optimal dual solution $(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\lambda})$ we use duality theory, following a similar construction to that used by Contreras et al. (2011) for the uncapacitated hub-location problem. Let us consider the complementary slackness conditions from both the primal and dual subproblems:

$$x_a^k(\alpha_{tail(a)} + \beta_a^k + \gamma_k) = 0 \qquad k \in \mathcal{K}, a \in \delta^-(u^k, 0) \tag{71}$$

$$x_a^k(-\alpha_{head(a)} + \beta_a^k) = 0 \qquad k \in \mathcal{K}, a \in \delta^+(\phi) \tag{72}$$

$$x_a^k(\alpha_{tail(a)} - \alpha_{head(a)} + \beta_a^k) = 0 \qquad k \in \mathcal{K}, a \in \mathcal{A}_k^* \tag{73}$$

$$z_s(\alpha_{succ(s)} - \alpha_s + \lambda_s) = 0 \qquad s \in \mathcal{S}_\mathcal{V} \tag{74}$$

$$y_s^-(1 + \alpha_s) = 0 \qquad s \in \mathcal{S}_\mathcal{V} \tag{75}$$

$$y_s^+(1 - \alpha_s) = 0 \qquad s \in \mathcal{S}_\mathcal{V} \tag{76}$$

$$\beta_a^k(x_a^k - Q_k \sum_{l \in \mathcal{B}_k} w_a^l \overline{\theta}_l) = 0 \qquad k \in \mathcal{K}, a \in \mathcal{A} \tag{77}$$

$$\lambda_s(C_{v(s)} - z_s) = 0 \qquad s \in \mathcal{S}_\mathcal{V}. \tag{78}$$

We use these conditions to derive an optimal dual solution from the optimal solution of the primal subproblem. We derive a partial dual solution by solving a reduced feasibility problem and extend it to a complete dual solution by simple inspection. We provide the details of this procedure in the Appendix.

4. Computational experience

In this section we report our computational experience with several families of instances and several different settings. To the best of our knowledge, no other method in the literature addresses the dynamic public bike-sharing balancing problem. Hence, we generated several random instances to test the performance of our methodology. We use an Intel Xeon E5462, 3.0-GHz processor with 16 GB of RAM. To solve the linear and integer linear programs we use CPLEX 12.3.

4.1. Instance generator

We generated a set of 120 instances with the following characteristics. We consider different numbers of stations, namely 25, 50, and 100, distributed in a plane with the x and y coordinates in the interval $[0, 60]$. The idea is to test the performance of our algorithm as the instance size increases. We consider a time horizon of 2 h, discretized with two different granularities. We consider 24 periods of 5 min each and 60 periods of 2 min each. The idea is to test the sensitivity of our algorithm to the granularity of the time discretization. The stations are either randomly distributed or clustered. In the former case, the points are randomly distributed in the plane and the stations are alternately pickup points or delivery points. In the latter case, the plane is subdivided into 9 clusters and 4 of these are randomly chosen to contain nodes. Inside a cluster all the stations are of the same type, either pickup or delivery. We consider a fixed fleet size of 5 vehicles for all instances. For each combination we generated 10 instances, for a total of 120 instances. The stations are considered either pickup or delivery points. In the former case, the demand f_s of a state $s = (v(s), t(s))$ is computed as follows. First, the real time $\widehat{t(s)}$ is considered (in minutes). If the length period is denoted l and $r(s)$ is a random integer between 1 and 5 for a node $s \in \mathcal{S}$, the demand at point s is

$$f_s = \lceil r(s) \times l \times \widehat{t(s)} \times (\widehat{t(s)} - 60) \times 10^{-3} \rceil. \tag{79}$$

If $v(s)$ is a delivery point, the demand is computed using the above formula but with opposite sign.

4.2. Algorithm settings

For the column generation algorithm, we perform a cleaning of the column pool every 50 iterations. We delete from the column pool all columns that have been nonbasic for 30 or more iterations. Other than that, we implemented a textbook version of the algorithm, i.e., stabilization methods and variable fixing procedures were not implemented; we leave these issues to future research.

For the Benders decomposition heuristic, we performed three preliminary tests to decide which columns to include in the set $\mathcal{B}$. First, we included all routes associated with the patterns generated so far. Second, we also included, among all the routes generated, those in which at least one associated pattern had zero reduced cost. Finally, we included all routes associated with basic patterns with strictly positive values. The third setting usually performed the best in terms of the balance between CPU time and solution quality. Also, we do not run the Benders algorithm to optimality. Instead, because the goal is to obtain a good feasible solution, we run it for 30 iterations. However, each time that we find a better solution, we perform 10 more iterations. Finally, by manipulating the coefficients of the objective function g in program (80)–(92) (see in the Appendix) we construct 2 dual solutions at each iteration. In the first case, the weights of the dual variables are set according to the coefficient of these variables in the original dual problem (63)–(70). In the second case, the coefficient in function g is set to -1 if the corresponding original coefficient is positive, and to 1 otherwise.

4.3. Computational results

To test the efficiency of our methodology, we designed and performed four different experiments.

In the first experiment, we compare the solution of problem (1)–(12) and the solution obtained by combining column generation with Benders decomposition (CG+BD). For the former, we use the CPLEX MIP solver with the default settings for a maximum of 30 min. In Table 1 we report the average results for the lower bounds (columns labeled LB) and upper bounds (columns labeled UB) obtained by the two methods (the label AFF indicates the arc-flow formulation and $CG+BD$ indicates column generation with Benders decomposition). For CG+BD, we also report the CPU times (columns labeled CPU, in seconds). These are decomposed into the average CPU time spent on column generation (column labeled CPU_{CG}, in seconds) and the average time spent on the Benders decomposition heuristic (column labeled CPU_{BD}, in seconds). The clustered and random instances are aggregated. Except for the smallest instances (25 stations and 24 periods of 5 min each), our method produces better lower and upper bounds than those achieved by the arc-flow model solved by a state-of-the-art commercial solver. Moreover, the solution is rapid (on average around 5 min for the largest instances considered). In many instances the arc-flow solver could not even solve the root-node relaxation. This demonstrates the robustness of our method, which scales well for these large instances. For the smallest set of instances considered (25 stations and 24 periods of 5 min each), the arc-flow formulation usually produces better bounds because of the application of flow-cover inequalities (that increase the value of the lower bound) and the heuristic performed by the solver at the root relaxation. In future research, the impact of flow-cover inequalities in CG+BD should be assessed.

$\|\mathcal{V}\|$	$\|\mathcal{T}\|$	AFF		CG+BD				
		LB	UB	LB	UB	CPU_{CG}	CPU_{BD}	CPU
25	24	**857.1**	**1177.2**	753.8	1231.25	0.5	14.4	14.9
25	60	732.2	2145.5	**858.3**	**1350.4**	4.5	52.5	57.0
50	24	2204.1	4030.1	**2351.5**	**3170.0**	1.9	23.6	25.5
50	60	0.0	6567.1	**2144.5**	**3192.6**	24.1	120.1	144.2
100	24	0.0	12701.7	**5486.3**	**6800.8**	7.3	72.0	79.3
100	60	0.0	13187.1	**5425.2**	**7171.2**	100.2	206.5	306.7

Table 1 **Comparison of arc-flow formulation and CG+BD**

In the second experiment, we compare the CG+BD results for random and clustered instances. In Table 2 we report the average lower bounds (columns labeled LB), average upper bounds (columns labeled UB), and average gaps (columns labeled gap). The gap is computed as $(z_{UB} - z_{LB})/z_{UB} \times 100$). The results show that the clustered instances are more rigid, in the sense that they usually accept worse solutions than the random instances do, but the lower bounds are stronger. We believe that this behavior is a consequence of the fact that for random instances the solutions have less structure. In the clustered case a vehicle must visit a zone with pickup nodes and then travel to a region with delivery nodes and vice versa. In the random case there is no clear pattern for the vehicle routes because the pickup and delivery nodes are mixed. Of course, real instances are likely to be clustered, because travel patterns are usually unidirectional, i.e., the service zone at peak hours can be divided into subzones, each of which is primarily a pickup region or primarily a delivery region.

$\|\mathcal{V}\|$	$\|\mathcal{T}\|$	Clustered			Random		
		LB	UB	*gap*	LB	UB	*gap*
25	24	982.8	1341.3	28.5	524.9	1121.2	53.6
25	60	1032.5	1447.8	29.4	684.1	1252.9	46.1
50	24	2887.3	3349.3	14.1	1815.8	2990.7	39.7
50	60	2674.9	3274.0	18.4	1614.0	3111.2	48.1
100	24	6457.2	7058.7	8.6	4515.5	6542.9	31.0
100	60	6571.7	7545.4	13.0	4278.7	6797.0	37.0

Table 2 Comparison of clustered and random instances

In the third experiment, we assess the sensitivity of our method to the granularity of the time discretization. In Table 3 we report the average lower bound (columns labeled *LB*), average upper bound (columns labeled *UB*), average gap (columns labeled *gap*, computed as before), and average CPU time (columns labeled *CPU*) of CG+BD for the two different granularities. It can be seen that the finer granularity has a negative impact on the CPU time and, for large instances, on the gap. We will not attempt to establish the optimal granularity of the time discretization; this should be considered in future research.

$\|\mathcal{V}\|$	24 Periods				60 Periods			
	LB	UB	*gap*	CPU	LB	UB	*gap*	CPU
25	753.8	1231.3	41.1	14.9	858.3	1350.4	37.8	57.0
50	2351.5	3170.0	26.9	25.5	2144.5	3192.6	33.2	144.2
100	5486.3	6800.8	19.8	79.3	5425.2	7171.2	25.0	306.7

Table 3 Algorithm sensitivity to time discretization

In the final experiment we evaluate the performance of a natural extension of our method. After solving the root-node relaxation of problem (13)–(21) and applying the Benders decomposition heuristic, we perform branch-and-price. We use OOBB (object-oriented branch-and-bound) as the branch-and-bound framework in the branch-and-price algorithm; it is a C++ library developed at CIRRELT. We branch on the vehicle-flow variables $w_a = \sum_{l\in\Omega} w_a^l \theta_l$, where w_a^l is a binary constant equal to 1 iff route l traverses arc a. Specifically, we branch on the arc-flow variable with the most fractional value. This branching rule allows us to solve the pricing problem at the internal nodes of the branching tree as a shortest path problem. At all nodes of the tree whose depth is a multiple of 5 we perform the Benders decomposition heuristic. Moreover, in the internal nodes the number of iterations of the Benders decomposition algorithm is reduced from 30 to 10. We chose these settings after performing a series of preliminary tests to find a compromise between the improvement provided by the heuristic versus the time spent in it. In Table 4 we report the average lower bounds (columns labeled LB), the average upper bounds (columns labeled UB), and the average number of nodes inspected after 10 min (column labeled N). Only a small gain is obtained by branching, especially for large instances; the bounds were not substantially improved during the branch-and-price. Future research should focus on the development of a more efficient algorithm capable of producing tighter gaps (better lower and upper bounds). The ultimate goal is to derive an exact solver capable of proving optimality for small and medium instances.

$\|\mathcal{V}\|$	$\|\mathcal{T}\|$	Root		10 min		
		LB	UB	LB	UB	N
25	24	753.8	1231.3	768.2	1198.1	434
25	60	858.3	1350.4	864.8	1321.1	120
50	24	2351.5	3170.0	2370.5	3120.0	228
50	60	2144.5	3192.6	2152.9	3179.7	34
100	24	5486.3	6800.8	5501.5	6747.4	136
100	60	5425.2	7171.2	5430.7	7156.8	13

Table 4 Impact of branch-and-price

5. Concluding remarks

We have introduced a dynamic public bike-sharing balancing problem. It arises in a PBS provider's daily operation of a fleet of trucks to transport bikes from full stations to stations with shortages. We formally defined the problem and proposed three mathematical formulations. To the best of our knowledge, this is the first time that this problem has been addressed from an OR perspective, and also the first time that a solution method has been developed for medium and large instances. Our methodology uses decomposition techniques to move the difficult variables and constraints into the subproblems, which can be solved efficiently. It also uses two different kinds of decomposition, Dantzig-Wolfe and Benders decomposition. To the best of our knowledge, this is the first time that these two approaches have been combined in a nested way for the solution of pickup-and-delivery problems. Our computational experience demonstrates that our methodology is effective for rapidly generating lower and upper bounds. However, the large gaps indicate that this is a challenging problem. Future research should focus on the development of algorithms capable of producing better lower and/or upper bounds. Future research could also investigate network-design decisions (the optimal location of the stations) or include stochasticity in the projected demand.

References

S. Anily and J. Bramel. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics*, 46(6):654–670, 1999.

S. Anily and R. Hassin. The swapping problem. *Networks*, 22:419–433, 1992.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

C. Bordenave, M. Gendreau, and G. Laporte. A branch-and-cut algorithm for the nonpreemptive swapping problem. *Naval Research Logistics*, 56(5):478–486, 2009.

C. Bordenave, M. Gendreau, and G. Laporte. Heuristics for the mixed swapping problem. *Computers & Operations Research*, 37(1):108 – 114, 2010.

D. Chemla, F. Meunier, and R. Wolfler-Calvo. Balancing the stations of a self-service bike hiring system. Working paper, 2011.

I. Contreras, J.-F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub-location. *Operations Research*, 59:1477–1490, 2011.

G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

L. dell'Olio, A. Ibeas, and J. Moura. Implementing bike-sharing systems. *Proceedings of the Institution of Civil Engineers: Municipal Engineer*, 164(2):89–101, 2011.

G. Erdogan, J.-F. Cordeau, and G. Laporte. A branch-and-cut algorithm for solving the non-preemptive capacitated swapping problem. *Discrete Applied Mathematics*, 158(15):1599–1614, 2010.

H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery travelling salesman problem. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization – Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 89–104. Springer Berlin / Heidelberg, 2003.

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

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

J.-R. Lin and T.-H. Yang. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):284–294, 2011.

T. Raviv, M. Tzur, and I. Forma. The static repositioning problem in a bike-sharing system. Working paper, 2011.

S. A. Shaheen, S. Guzman, and H. Zhang. Bike sharing in Europe, the America and Asia: Past, present and future. In *Transportation Research Board 89th Annual Meeting*, Washington, D.C., 2010.

P. Vogel and D. Mattfeld. Strategic and operational planning of bike-sharing systems by data mining - a case study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6971 LNCS:127–141, 2011. cited By (since 1996) 0.

S. Wang, J. Zhang, L. Liu, and Z.-Y. Duan. Bike-sharing-a new public transportation mode: State of the practice & prospects. pages 222–225, 2010. cited By (since 1996) 0.

F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Industrial Engineering*, 56(4):1642–1648, 2009.

Appendix. Proofs of propositions

In this Appendix we prove the propositions in the text.

Proof of Proposition 1 Without loss of generality, assume that s is the latest node in time for which a pathological situation occurs (ties are broken arbitrarily). In other words, no pathological situations occur for nodes $s' \in \mathcal{S}_\mathcal{V}$ such that $t(s') > t(s)$. Let $(\overline{x}', \overline{y}', \overline{z}', \overline{w}')$ be a copy of $(\overline{x}, \overline{y}, \overline{z}, \overline{w})$. Let us consider each pathological case separately.

1. Suppose first that $\overline{y}_s^- > 0$. We set $\overline{y}'^-_s = 0$. This creates an imbalance in the system of $\overline{y}_s^-$ units. This must be corrected either in $\overline{z}'_s$ or in $\sum_{k\in\mathcal{K}} \sum_{a\in\delta^-(s)} \overline{x}'^k_a$. This is possible because $0 < q(s) < C_{v(s)} + \sum_{k\in\mathcal{K}} \sum_{a\in\delta^+(s)} \overline{x}'^k_a$. This adjustment creates an imbalance in the arriving nodes. To compensate for this at a node s' such that $t(s') > t(s)$ either the new bikes must leave the node using outgoing arcs, if possible, or we increase the value of variable $\overline{y}'^-_{s'}$. In total, at most $\overline{y}_s^-$ units of flow must be added in the future, and so $\sum_{s\in\mathcal{S}_\mathcal{V}} y'^+_s + y'^-_s \leq \sum_{s\in\mathcal{S}_\mathcal{V}} y_s^+ + y_s^-$. Suppose now that $\overline{y}_s^+ > 0$. Instead of inserting bikes into the system, we must remove bikes from it. When removing bikes from the outgoing arcs of node s (which is again possible because $0 < q(s) < C_{v(s)} + \sum_{k\in\mathcal{K}} \sum_{a\in\delta^+(s)} \overline{x}'^k_a$), we create imbalances, let us say in a node s' such that $t('s) > t(s)$. This again must be corrected either by removing flow from the outgoing arcs of this node (if possible) or by increasing the value of $\overline{y}'^+_s$. Again, we will obtain $\sum_{s\in\mathcal{S}_\mathcal{V}} y'^+_s + y'^-_s \leq \sum_{s\in\mathcal{S}_\mathcal{V}} y_s^+ + y_s^-$.

2. In this case, we set $\overline{y}'^+_s = -q(s)$. To correct the ingoing and outgoing flows at this node, we have to remove $\overline{y}_s^+ + q(s)$ units of flow from the outgoing arcs at node s, making the outgoing flow zero. This adjustment will have an impact on future nodes, namely s' such that $t(s') > t(s)$. To compensate for the missing flow, we must either decrease the outgoing flow, if possible, or increase the value of variable $\overline{y}'^+_{s'}$. Again, the impact of these actions in the future is at most $\overline{y}_s^+ + q(s)$ units, so we will obtain $\sum_{s\in\mathcal{S}_\mathcal{V}} y'^+_s + y'^-_s \leq \sum_{s\in\mathcal{S}_\mathcal{V}} y_s^+ + y_s^-$.

3. The reasoning for this case is analogous to that applied in the previous cases. For the sake of brevity we omit it.

□

Appendix. Obtaining a dual solution from the complementary slackness conditions

Let $(\overline{x}, \overline{y^+}, \overline{y^-}, \overline{z})$ be an optimal solution for the primal subproblem (55)–(62). The feasibility problem used to derive the remaining dual variables in Section 3.2.3 is as follows:

$$\min \quad g(\alpha, \beta, \gamma, \lambda) \tag{80}$$

s. t.

$$\alpha_{tail(a)} + \beta_a^k + \gamma_k = 0 \qquad k \in \mathcal{K}, a \in \delta^-(u^k, 0), \overline{x_a^k} > 0 \tag{81}$$

$$-\alpha_{head(a)} + \beta_a^k = 0 \qquad k \in \mathcal{K}, a \in \delta^+(\phi), \overline{x_a^k} > 0 \tag{82}$$

$$\alpha_{tail(a)} - \alpha_{head(a)} + \beta_a^k = 0 \qquad k \in \mathcal{K}, a \in \mathcal{A}_k^*, \overline{x_a^k} > 0 \tag{83}$$

$$\alpha_{succ(s)} - \alpha_s + \lambda_s = 0 \qquad s \in \mathcal{S}_\mathcal{V}, \overline{z_s} > 0 \tag{84}$$

$$\alpha_s = -1 \qquad s \in \mathcal{S}_\mathcal{V}, \overline{y_s^-} > 0 \tag{85}$$

$$\alpha_s = 1 \qquad s \in \mathcal{S}_\mathcal{V}, \overline{y_s^+} > 0 \tag{86}$$

$$\beta_a^k = 0 \qquad k \in \mathcal{K}, a \in \mathcal{A}, \overline{x_a^k} < Q_k \sum_{l \in \mathcal{B}_k} w_a^l \overline{\theta}_l \tag{87}$$

$$\lambda_s = 0 \qquad s \in \mathcal{S}_\mathcal{V}, \overline{z_s} < C_{v(s)} \tag{88}$$

$$\alpha_{succ(s)} - \alpha_s + \lambda_s \leq 0 \qquad s \in \mathcal{S}_\mathcal{V} \tag{89}$$

$$-1 \leq \alpha_s \leq 1 \qquad s \in \mathcal{S}_\mathcal{V} \tag{90}$$

$$\beta, \lambda \leq 0 \tag{91}$$

$$\gamma \text{ unrestricted.} \tag{92}$$

Note that this feasibility problem can be restricted to contain only those variables appearing in some constraint. In particular, it is possible to discard a priori all variables β_a^k such that $Q_k \sum_{l \in \mathcal{B}_k} w_a^l \overline{\theta}_l = 0$, which includes all the arcs not used in any route. The function g can be zero, but it can also give weights to the dual variables to obtain strong Benders cuts. Once this problem is solved, the partial dual solution must be completed to find the values of the remaining β variables. We apply the following procedure. For each arc $a \in \mathcal{A}$ and vehicle $k \in \mathcal{K}$ such that $Q_k \sum_{l \in \mathcal{B}_k} w_a^l \overline{\theta}_l = 0$, we fix the corresponding variable β_a^k to the largest possible value (i.e., closest to zero) that satisfies the corresponding constraint in (64)–(66). In particular, if the corresponding constraint is already satisfied then we set $\beta_a^k = 0$. Note also that the overall Benders method can be accelerated by adding several cuts at each iteration instead of just one. In this case, this is not expensive since it involves adjusting at each time the coefficients in the objective function $g(\alpha, \beta, \gamma, \lambda)$ and re-solving problem (80)–(92). We have found that solving this problem twice provides a reasonable balance between the extra time needed and the faster convergence of the overall algorithm in terms of the number of iterations.

Appendix. Disaggregated results

In this Appendix we report the results of our algorithm for all the instances. In Tables 5–7 we report the disaggregated results of our methodology. In these tables, we report the lower bounds (columns labeled LB), upper bounds (columns labeled UB), and CPU time (columns labeled CPU, in seconds). We also report the CPU times for the column generation procedure (columns labeled CPU_{CG}, in seconds) and the Benders decomposition heuristic (columns labeled CPU_{BD}, in seconds). In Tables 8–10 we report (columns labeled $\#N$) the number of nodes inspected during the branch-and-bound when running the branch-and-price for 10 min. The name of each instance indicates the number of stations, the number of periods, and if the instances are clustered or randomly distributed. For example, cmr-10x5x24-1r indicates an instance with 10 stations, 5 vehicles, 24 time periods, and random distribution of the stations.

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU
cmr-25x5x24-1c	1030.6	1264	0.3	15.2	15.5
cmr-25x5x24-1r	583.1	919	0.4	13.7	14.1
cmr-25x5x24-2c	1102.3	1390	0.4	14.5	14.9
cmr-25x5x24-2r	369.8	996	0.7	54.8	55.5
cmr-25x5x24-3c	554.2	1027	0.9	17.1	17.9
cmr-25x5x24-3r	831.3	1298	0.4	4.5	4.9
cmr-25x5x24-4c	1364.0	1541	0.1	3.8	3.9
cmr-25x5x24-4r	827.2	1261	0.3	7.3	7.6
cmr-25x5x24-5c	683.7	1139	0.6	20.1	20.7
cmr-25x5x24-5r	193.5	1018	0.4	3.1	3.5
cmr-25x5x24-6c	1018.1	1411	0.5	11.5	12.0
cmr-25x5x24-6r	538.3	1206	0.5	5.6	6.2
cmr-25x5x24-7c	543.9	994	1.0	13.3	14.4
cmr-25x5x24-7r	583.9	1094	0.7	5.4	6.2
cmr-25x5x24-8c	1432.6	1701	0.4	8.2	8.6
cmr-25x5x24-8r	425.8	1091	0.4	20.9	21.3
cmr-25x5x24-9c	780.7	1305	0.8	36.6	37.4
cmr-25x5x24-9r	385.8	972	0.3	8.3	8.6
cmr-25x5x24-10c	1317.4	1641	0.3	9.5	9.8
cmr-25x5x24-10r	510.2	1357	0.5	14.3	14.8
cmr-25x5x60-1c	1334.3	1770	5.3	22.1	27.3
cmr-25x5x60-1r	746.1	1303	2.5	24.5	27.0
cmr-25x5x60-2c	1390.0	1621	1.7	28.5	30.1
cmr-25x5x60-2r	623.9	1259	3.2	13.1	16.3
cmr-25x5x60-3c	593.6	1067	9.8	145.9	155.7
cmr-25x5x60-3r	457.8	1061	7.0	177.4	184.4
cmr-25x5x60-4c	1045.3	1419	4.2	59.4	63.6
cmr-25x5x60-4r	552.4	978	4.5	46.3	50.7
cmr-25x5x60-5c	1149.9	1540	4.9	94.3	99.3
cmr-25x5x60-5r	273.2	1132	6.7	47.6	54.3
cmr-25x5x60-6c	906.3	1502	7.5	42.9	50.4
cmr-25x5x60-6r	681.9	1165	3.7	50.7	54.4
cmr-25x5x60-7c	1027.9	1569	4.1	59.7	63.8
cmr-25x5x60-7r	882.5	1462	2.7	35.0	37.6
cmr-25x5x60-8c	692.4	1110	7.2	91.1	98.3
cmr-25x5x60-8r	697.2	1396	4.6	13.1	17.7
cmr-25x5x60-9c	1186.6	1628	3.0	34.2	37.2
cmr-25x5x60-9r	988.1	1485	2.2	29.4	31.6
cmr-25x5x60-10c	999.2	1252	2.6	15.3	17.8
cmr-25x5x60-10r	938.2	1288	3.1	19.4	22.6

Table 5 Results for instances containing 25 stations

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU
cmr-50x5x24-1c	2203.2	2908	2.4	18.7	21.1
cmr-50x5x24-1r	1361.5	2688	1.8	43.9	45.7
cmr-50x5x24-2c	3448.5	3744	1.0	14.9	15.9
cmr-50x5x24-2r	2427.8	3660	1.6	17.9	19.5
cmr-50x5x24-3c	2971.2	3273	1.7	18.1	19.8
cmr-50x5x24-3r	1154.6	2641	1.9	32.9	34.8
cmr-50x5x24-4c	2319.4	2961	2.4	16.9	19.3
cmr-50x5x24-4r	2004.1	3021	2.0	48.1	50.1
cmr-50x5x24-5c	2731.1	3416	2.4	24.4	26.8
cmr-50x5x24-5r	1594.3	2957	2.6	33.7	36.3
cmr-50x5x24-6c	2929.7	3410	4.1	24.4	28.5
cmr-50x5x24-6r	1683.9	2620	1.8	30.8	32.6
cmr-50x5x24-7c	3089.9	3452	1.4	9.9	11.4
cmr-50x5x24-7r	2043.3	3116	1.6	25.1	26.7
cmr-50x5x24-8c	2848.8	3436	1.7	12.9	14.5
cmr-50x5x24-8r	2007.9	3257	1.1	23.8	24.9
cmr-50x5x24-9c	2871.6	3300	2.0	20.6	22.6
cmr-50x5x24-9r	1861.3	2926	1.7	18.0	19.7
cmr-50x5x24-10c	3459.1	3593	1.1	15.7	16.8
cmr-50x5x24-10r	2019.7	3021	1.7	21.4	23.1
cmr-50x5x60-1c	3209.2	3517	12.1	48.7	60.7
cmr-50x5x60-1r	1687.5	3269	13.1	52.9	65.9
cmr-50x5x60-2c	2831.5	3635	45.2	115.5	160.6
cmr-50x5x60-2r	2029.5	3337	12.7	92.8	105.5
cmr-50x5x60-3c	2750.8	3303	18.7	92.7	111.3
cmr-50x5x60-3r	1276.0	2884	11.3	127.6	138.9
cmr-50x5x60-4c	2691.6	3178	30.4	108.5	138.9
cmr-50x5x60-4r	1339.7	3357	23.1	99.2	122.3
cmr-50x5x60-5c	2419.6	3280	48.4	374.6	423.0
cmr-50x5x60-5r	1143.0	3237	30.2	94.5	124.7
cmr-50x5x60-6c	3170.3	3648	20.6	66.4	87.0
cmr-50x5x60-6r	2024.4	3077	15.4	115.8	131.2
cmr-50x5x60-7c	2182.4	2743	28.1	77.1	105.2
cmr-50x5x60-7r	1692.7	3118	14.2	115.0	129.2
cmr-50x5x60-8c	2311.3	3028	27.0	31.4	58.4
cmr-50x5x60-8r	1324.0	2719	18.2	143.9	162.0
cmr-50x5x60-9c	2536.6	3007	39.8	53.9	93.7
cmr-50x5x60-9r	1863.8	2975	11.3	156.9	168.2
cmr-50x5x60-10c	2646.0	3401	46.9	68.1	114.9
cmr-50x5x60-10r	1759.1	3139	15.9	366.9	382.8

Table 6 Results for instances containing 50 stations

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU
cmr-100x5x24-1c	6437.5	7084	16.3	44.8	61.1
cmr-100x5x24-1r	4812.2	6797	11.3	83.2	94.4
cmr-100x5x24-2c	6215.8	6566	6.5	22.9	29.4
cmr-100x5x24-2r	4104.7	6441	7.4	59.2	66.6
cmr-100x5x24-3c	6070.1	6930	5.9	20.1	26.0
cmr-100x5x24-3r	3965.2	6190	7.8	67.3	75.2
cmr-100x5x24-4c	7581.0	7894	4.7	21.3	26.0
cmr-100x5x24-4r	5085.3	6811	4.4	35.0	39.4
cmr-100x5x24-5c	6725.7	6931	3.1	17.5	20.6
cmr-100x5x24-5r	4951.4	7075	2.9	28.8	31.7
cmr-100x5x24-6c	6828.6	7660	7.3	33.0	40.2
cmr-100x5x24-6r	4573.0	6342	5.4	41.5	46.9
cmr-100x5x24-7c	5749.3	6322	7.8	55.4	63.2
cmr-100x5x24-7r	4496.8	6331	8.5	45.9	54.4
cmr-100x5x24-8c	6324.2	6886	5.5	22.7	28.2
cmr-100x5x24-8r	4095.1	6764	5.6	305.0	310.7
cmr-100x5x24-9c	5281.1	6477	14.7	203.7	218.4
cmr-100x5x24-9r	4518.6	6364	5.6	239.0	244.6
cmr-100x5x24-10c	7358.6	7837	11.0	29.3	40.3
cmr-100x5x24-10r	4552.5	6314	3.7	64.6	68.3
cmr-100x5x60-1c	5938.7	7051	115.7	114.2	229.8
cmr-100x5x60-1r	3641.3	6379	66.4	179.1	245.4
cmr-100x5x60-2c	6673.0	7932	210.6	66.5	277.1
cmr-100x5x60-2r	4336.3	6296	19.2	65.3	84.5
cmr-100x5x60-3c	6132.0	7729	273.0	603.6	876.6
cmr-100x5x60-3r	3701.0	6918	86.6	328.7	415.3
cmr-100x5x60-4c	5126.7	6219	125.7	57.7	183.3
cmr-100x5x60-4r	5788.2	7557	25.0	452.7	477.7
cmr-100x5x60-5c	6792.8	7511	219.9	109.3	329.2
cmr-100x5x60-5r	3846.0	7156	97.6	294.9	392.5
cmr-100x5x60-6c	7301.9	8106	129.5	228.1	357.6
cmr-100x5x60-6r	4151.6	6251	35.2	128.5	163.7
cmr-100x5x60-7c	7435.7	7875	48.0	302.1	350.1
cmr-100x5x60-7r	4502.8	6404	21.9	129.8	151.7
cmr-100x5x60-8c	6777.4	7883	179.4	296.7	476.2
cmr-100x5x60-8r	4219.7	6823	37.8	277.4	315.2
cmr-100x5x60-9c	7036.5	7704	71.5	158.8	230.3
cmr-100x5x60-9r	4597.7	7215	63.7	93.8	157.6
cmr-100x5x60-10c	6502.2	7444	130.1	141.6	271.7
cmr-100x5x60-10r	4002.2	6971	46.7	101.9	148.7

Table 7 Results for instances containing 100 stations

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU	#N
cmr-25x5x24-1c	1038.4	1244	469.8	130.7	600.5	792
cmr-25x5x24-1r	602.9	898	473.5	129.5	603.1	470
cmr-25x5x24-2c	1108.6	1368	495.6	105.9	601.5	513
cmr-25x5x24-2r	381.2	962	478.1	123.1	601.2	314
cmr-25x5x24-3c	560.5	988	476.5	125.6	602.1	278
cmr-25x5x24-3r	847.0	1266	531.7	70.3	602.0	440
cmr-25x5x24-4c	1369.3	1514	505.3	95.8	601.1	1009
cmr-25x5x24-4r	845.7	1236	534.4	67.9	602.3	316
cmr-25x5x24-5c	694.4	1065	463.8	136.9	600.7	332
cmr-25x5x24-5r	213.3	963	564.9	36.8	601.7	298
cmr-25x5x24-6c	1030.9	1378	508.5	91.7	600.1	422
cmr-25x5x24-6r	553.8	1141	511.1	90.9	602.0	321
cmr-25x5x24-7c	555.8	977	524.5	77.9	602.5	192
cmr-25x5x24-7r	596.4	1094	536.0	66.8	602.8	314
cmr-25x5x24-8c	1445.0	1678	454.3	145.9	600.2	1020
cmr-25x5x24-8r	448.3	1080	495.8	105.3	601.1	286
cmr-25x5x24-9c	787.2	1247	482.2	121.2	603.4	299
cmr-25x5x24-9r	408.7	972	528.0	73.6	601.6	272
cmr-25x5x24-10c	1329.6	1611	526.4	75.2	601.6	568
cmr-25x5x24-10r	546.4	1279	510.7	94.8	605.5	232
cmr-25x5x60-1c	1340.6	1729	499.3	104.2	603.4	126
cmr-25x5x60-1r	759.2	1260	510.8	106.3	617.1	84
cmr-25x5x60-2c	1390.0	1621	522.8	80.5	603.3	190
cmr-25x5x60-2r	633.2	1195	531.0	71.0	602.0	152
cmr-25x5x60-3c	597.1	1057	442.8	168.4	611.1	76
cmr-25x5x60-3r	465.2	1010	329.2	291.7	620.9	33
cmr-25x5x60-4c	1049.4	1397	433.6	170.1	603.7	168
cmr-25x5x60-4r	556.0	978	533.2	80.0	613.2	80
cmr-25x5x60-5c	1156.1	1517	365.9	237.2	603.1	160
cmr-25x5x60-5r	283.7	1113	533.0	70.7	603.6	60
cmr-25x5x60-6c	916.6	1460	431.9	174.4	606.3	92
cmr-25x5x60-6r	688.4	1165	490.3	114.1	604.4	127
cmr-25x5x60-7c	1034.6	1544	475.1	125.3	600.4	92
cmr-25x5x60-7r	889.9	1426	474.7	131.4	606.1	122
cmr-25x5x60-8c	696.1	1045	447.1	159.5	606.6	108
cmr-25x5x60-8r	704.0	1363	548.2	60.4	608.6	85
cmr-25x5x60-9c	1196.3	1578	447.2	157.0	604.2	152
cmr-25x5x60-9r	996.8	1482	495.1	109.2	604.4	108
cmr-25x5x60-10c	1001.6	1236	484.5	117.6	602.1	215
cmr-25x5x60-10r	942.2	1246	517.2	84.1	601.3	172

Table 8 Results for instances containing 25 stations after 10 min

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU	#N
cmr-50x5x24-1c	2204.1	2868	492.5	108.1	600.6	208
cmr-50x5x24-1r	1386.4	2556	483.0	123.5	606.4	117
cmr-50x5x24-2c	3474.7	3723	485.1	116.5	601.6	460
cmr-50x5x24-2r	2468.5	3594	499.5	100.7	600.3	124
cmr-50x5x24-3c	2972.3	3251	500.1	104.4	604.5	327
cmr-50x5x24-3r	1190.9	2542	480.4	120.9	601.2	115
cmr-50x5x24-4c	2322.1	2884	509.3	100.2	609.4	148
cmr-50x5x24-4r	2034.9	3003	452.7	147.3	600.0	190
cmr-50x5x24-5c	2752.4	3370	452.6	152.4	605.0	194
cmr-50x5x24-5r	1636.9	2900	490.6	116.1	606.7	107
cmr-50x5x24-6c	2938.4	3384	475.9	126.5	602.4	172
cmr-50x5x24-6r	1710.3	2586	447.8	154.4	602.2	194
cmr-50x5x24-7c	3094.8	3424	457.6	145.2	602.8	455
cmr-50x5x24-7r	2067.6	3068	446.3	158.2	604.5	230
cmr-50x5x24-8c	2855.5	3424	511.3	92.2	603.5	192
cmr-50x5x24-8r	2025.8	3171	462.1	139.1	601.1	189
cmr-50x5x24-9c	2875.1	3217	420.0	181.2	601.2	285
cmr-50x5x24-9r	1888.9	2897	488.1	114.9	603.0	196
cmr-50x5x24-10c	3465.1	3584	463.8	138.3	602.1	469
cmr-50x5x24-10r	2044.9	2954	491.6	117.4	608.9	194
cmr-50x5x60-1c	3213.1	3517	438.6	168.4	607.0	112
cmr-50x5x60-1r	1696.2	3234	549.7	112.3	662.0	27
cmr-50x5x60-2c	2833.1	3629	410.9	198.8	609.7	24
cmr-50x5x60-2r	2039.3	3337	435.2	167.4	602.6	35
cmr-50x5x60-3c	2753.8	3303	427.7	180.2	607.9	39
cmr-50x5x60-3r	1293.8	2819	327.7	272.7	600.4	32
cmr-50x5x60-4c	2693.4	3195	404.2	224.6	628.8	31
cmr-50x5x60-4r	1357.2	3357	468.9	160.4	629.4	22
cmr-50x5x60-5c	2423.4	3280	205.4	403.9	609.3	9
cmr-50x5x60-5r	1157.0	3215	483.1	196.6	679.7	14
cmr-50x5x60-6c	3177.5	3609	391.6	209.3	600.9	51
cmr-50x5x60-6r	2041.3	3077	354.8	246.0	600.7	32
cmr-50x5x60-7c	2185.5	2695	436.4	166.6	603.0	43
cmr-50x5x60-7r	1709.0	3118	413.7	191.6	605.3	28
cmr-50x5x60-8c	2314.3	3028	566.4	84.1	650.5	36
cmr-50x5x60-8r	1333.2	2719	381.3	229.9	611.2	26
cmr-50x5x60-9c	2540.4	2997	437.0	181.6	618.6	41
cmr-50x5x60-9r	1875.3	2975	417.1	186.2	603.3	33
cmr-50x5x60-10c	2651.2	3369	423.7	176.5	600.2	20
cmr-50x5x60-10r	1769.3	3120	177.2	425.0	602.2	20

Table 9 Results for instances containing 50 stations after 10 min

Instance	LB	UB	CPU_{CG}	CPU_{BD}	CPU	#N
cmr-100x5x24-1c	6437.5	7056	431.2	170.3	601.5	149
cmr-100x5x24-1r	4840.4	6765	485.4	127.9	613.3	39
cmr-100x5x24-2c	6215.8	6540	397.0	204.2	601.2	216
cmr-100x5x24-2r	4140.7	6453	469.9	130.4	600.2	49
cmr-100x5x24-3c	6072.6	6827	402.3	202.4	604.7	108
cmr-100x5x24-3r	3988.1	6089	451.9	154.2	606.1	39
cmr-100x5x24-4c	7586.0	7870	421.3	181.4	602.7	345
cmr-100x5x24-4r	5131.6	6786	435.6	166.2	601.8	173
cmr-100x5x24-5c	6725.7	6913	461.1	139.3	600.4	376
cmr-100x5x24-5r	4989.8	6982	440.0	160.7	600.6	136
cmr-100x5x24-6c	6840.3	7609	450.3	160.2	610.6	98
cmr-100x5x24-6r	4578.1	6229	453.5	150.5	604.0	142
cmr-100x5x24-7c	5749.3	6255	447.1	156.1	603.2	115
cmr-100x5x24-7r	4516.0	6256	320.3	282.3	602.6	68
cmr-100x5x24-8c	6329.3	6894	462.1	144.3	606.4	140
cmr-100x5x24-8r	4127.2	6760	242.2	365.7	608.0	80
cmr-100x5x24-9c	5289.8	6433	363.1	237.0	600.1	42
cmr-100x5x24-9r	4535.7	6285	275.6	333.0	608.6	60
cmr-100x5x24-10c	7358.6	7790	409.9	191.4	601.3	176
cmr-100x5x24-10r	4576.5	6156	429.2	173.4	602.6	165
cmr-100x5x60-1c	5942.2	7051	441.6	167.2	608.9	10
cmr-100x5x60-1r	3652.2	6379	413.7	253.9	667.6	12
cmr-100x5x60-2c	6673.0	7932	709.1	83.2	792.3	5
cmr-100x5x60-2r	4354.5	6247	388.3	212.9	601.2	33
cmr-100x5x60-3c	6132.0	7729	273.1	603.8	876.9	1
cmr-100x5x60-3r	3725.4	6918	290.7	360.1	650.8	5
cmr-100x5x60-4c	5126.7	6186	492.2	118.2	610.4	13
cmr-100x5x60-4r	5793.3	7557	99.2	502.1	601.3	9
cmr-100x5x60-5c	6794.2	7487	485.8	155.6	641.4	10
cmr-100x5x60-5r	3846.0	7156	286.0	328.5	614.5	6
cmr-100x5x60-6c	7303.0	8096	351.8	303.4	655.2	10
cmr-100x5x60-6r	4163.6	6194	369.6	233.0	602.5	36
cmr-100x5x60-7c	7438.4	7875	189.7	421.6	611.2	18
cmr-100x5x60-7r	4518.4	6404	355.6	250.6	606.2	39
cmr-100x5x60-8c	6777.7	7883	287.2	341.8	629.0	6
cmr-100x5x60-8r	4225.3	6789	263.9	363.7	627.6	10
cmr-100x5x60-9c	7036.5	7644	124.0	476.7	600.6	6
cmr-100x5x60-9r	4607.6	7215	498.9	150.4	649.3	11
cmr-100x5x60-10c	6502.7	7444	507.1	182.0	689.1	9
cmr-100x5x60-10r	4002.2	6950	403.1	224.9	627.9	19

Table 10 Results for instances containing 100 stations after 10 min