Computers & Operations Research 38 (2011) 1054–1065

Contents lists available at ScienceDirect

Computers & Operations Research

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

Heuristic and exact algorithms for a min–max selective vehicle routing problem

Cristiano Arbex Valle, Leonardo Conegundes Martinez, Alexandre Salles da Cunha *,[1], Geraldo R. Mateus [2]

Departamento de Ciência da Computação Universidade Federal de Minas Gerais Belo Horizonte, Minas Gerais, Brazil

ARTICLE INFO

Available online 14 October 2010

Keywords:
Vehicle routing problem
Branch-and-cut
Local branching
Heuristics

ABSTRACT

In this work, we investigate a vehicle routing problem where not all clients need to be visited and the goal is to minimize the longest vehicle route. We propose two exact solution approaches for solving the problem: a Branch-and-cut (BC) algorithm and a Local Branching (LB) method that uses BC as its inner solver. Our computational experience indicates that, in practice, the problem is difficult to solve, mainly when the number of vehicles grows. In addition to the exact methods, we present a heuristic that relies on GRASP and on the resolution of a restricted integer program based on a set covering reformulation for the problem. The heuristic was capable of significantly improving the best solutions provided by BC and LB, in one tenth of the times taken by them to achieve their best upper bounds.

1. Introduction

Vehicle routing problem (VRP) is the generic name assigned to a large class of problems that seek the optimal delivery and/or collection of goods to final clients through the efficient use of a fleet of vehicles. VRP is one of the most studied problems in combinatorial optimization, due both to its difficulty and relevance in practice.

Given a set of vehicles, a set of clients (or customers) and distances between the clients, the non-capacitated VRP seeks a set of routes (one for each vehicle) such that each client is visited by a vehicle exactly once and the total distance traveled is minimized. In its capacitated version, an unsplitable demand of a certain good is assigned to each client and vehicle capacities restrict the amount of goods that can be shipped in any feasible route.

VRP generalizes the Traveling Salesman Problem (TSP), but seems to be much harder to solve. This appears to be true since the literature routinely reports TSP instances with thousands of clients being solved, whereas for the VRP case, the best known solution algorithms (see [1,2], for instance) can solve instances with only few hundred clients.

The roots of the problem trace back to the pioneering work of Dantzig and Ramser [3], who first formulated and solved a VRP concerning the optimal distribution of gasoline to 12 gas stations. Later, Clarke and Wright [4] proposed a heuristic that considerably improved on the approach in [3]. In the past decades, was observed a very fast growing number of studies on solution techniques, both exact [5–8,1,2] and heuristic [9–13]. For an updated review of the most important contributions on the VRP literature, we suggest the excellent surveys of Toth and Vigo [14], Golden et al. [15] and Laporte [16].

In practice, several variants of the problem exist, since, quite often, the operating rules and constraints found in real applications are more complex and diversified [16]. Some variants that received the greatest attention by practioners and by the scientific community are: (i) the Distance-constrained VRP [17,18], where each route must have a length not exceeding a given threshold value; (ii) the VRP with time windows [19–21], where each client must be visited within an associated time interval; (iii) the VRP with pickup and delivery [22], where a fleet of heterogeneous vehicles based at multiple depots must satisfy transportation requests, each one defined by a pickup and a corresponding delivery point and by a demand of goods to be transported from one to the other; and (iv) the VRP with backhauls [23], the version in which the vehicle may also pickup and deliver during the same trip, but in any feasible route, the subset of clients to whom a given demand of goods is assigned (the linehaul clients) must be visited after the subset of clients from whom the goods must be picked up (the backhaul clients). In this paper, we introduce a (uncapacitated) VRP that models an application in Wireless Sensor Networks (WSN) described in [24]. It differs from other variants in the VRP literature by two major aspects:

- Not all clients need to be visited by a vehicle. Differently, each client that is not visited must be *close enough* to a client that is visited.

* Corresponding author.
E-mail addresses: arbex@dcc.ufmg.br (C.A. Valle), leocm@dcc.ufmg.br (L.C. Martinez), acunha@dcc.ufmg.br (A.S. da Cunha), mateus@dcc.ufmg.br (G.R. Mateus).

[1] Alexandre Salles da Cunha was partially funded by CNPq grant 302276/2009-2, by FAPEMIG grant 14016*1 and by FAPEMIG/PRONEX APQ-01201-09.
[2] Geraldo Robson Mateus was partially funded by CNPq grant 55.0790/2007-1 and by FAPEMIG/PRONEX APQ-01201-09.

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.10.010

- The goal is not to minimize the total distance traveled. Instead of that, the objective is to minimize the longest route.

Due to the features quoted above, we name this routing problem as the Min-Max Selective Vehicle Routing Problem (MMSVRP). Roughly speaking, the proposed optimization problem arises as an effort to deal with two conflicting objectives in WSN: to reduce energy consumption in sensor nodes and to lower message delays. Further details about the application in WSN that MMSVRP relates to could be found in Valle et al. [24].

To the best of our knowledge, MMSVRP has never been studied before. The problem we found in the literature that is the closest to MMSVRP is the VRP discussed by Glaab [25], which arises in the design of a semi-automatic system for cutting leather skins. As in MMSVRP, the fleet size is fixed and the goal is to minimize the longest route length. However, MMSVRP differs from that routing problem in two aspects: in [25], all clients need to be visited and each vehicle has a different depot. To provide feasible solutions to real world instances with up to 214 vertices, Glaab [25] designed insertion and greedy algorithms. On the dual side, Glaab replaced the min–max objective function by the average length of the tours and then used one-tree and matching relaxations to provide lower bounds to the problem.

Min–max routing problems are not very common in the literature. One of the few additional references we are aware of is the min–max K-vehicle windy rural postman problem, introduced by Benavent et al. [26]. Given a graph $G=(V,E)$ with two non-negative costs c_{ij} and c_{ji} assigned to each edge $\{i,j\}$ of E (corresponding to the costs of transversing the edge from i to j and from j to i), a subset of edges $E_R \subseteq E$, a fleet of K vehicles and a common depot, the K-vehicle windy rural postman problem is an arc routing problem that consists in finding a set of K tours for the vehicles such that each one starts and ends at the depot and each edge in E_R is transversed at least once by a vehicle. The goal is to find a set of K tours where the length of the longest is minimized. In [26], the authors suggested an integer programming formulation, conducted a polyhedral study of the underlying polytope, characterizing facet defining inequalities, and proposed a branch-and-cut algorithm for solving the problem.

Another combinatorial optimization problem that also relates to MMSVRP is the capacitated m-ring star problem [27,28] (CmRSP). For the CmRSP, the goal is to find a minimal cost set of m rings such that either a node is included in a ring or else it is assigned to a node in the ring. The objective function considers the construction cost of the ring as well as the cost of assigning a vertex to a node in one of the rings. In this sense, MMSVRP differs from CmRSP in three basic aspects: capacity constraints are not imposed on the routes in MMSVRP, there are no assignment costs in MMSVRP and the objective function in CmRSP does not have the min–max nature.

In this work, we extend the results and the algorithms presented in a conference version of this paper [29]. We investigate two exact algorithms for MMSVRP: a branch-and-cut method and a Local Branching procedure that uses BC as its inner solver. Our computational experience with these algorithms suggest that, in practice, MMSVRP is difficult to solve to proven optimality. Motivated by that, we propose a fast heuristic that relies on GRASP [30] and on the resolution of a restricted integer program based on a set covering reformulation for MMSVRP. The heuristic was capable of significantly improving on the best upper bounds for MMSVRP, with little computational effort.

The rest of the paper is organized as follows. In Section 2, we define the problem and introduce most of the notation used through the paper. In Sections 3 and 4 we respectively present the exact and heuristic algorithms devised for MMSVRP. In Section 5, we report our computational experience with the proposed algorithms. We conclude the paper in Section 6, indicating directions for future investigation.

2. Problem definition

In this section, we present the notation used through the paper and define MMSVRP. To that aim, let us consider the following parameters:

- $\mathcal{K}=\{1,\ldots,K\}$: a set of identical non-capacitated vehicles.
- $G=(V,E)$: an undirected graph, with set of vertices $V=\{1,\ldots,n\}$ and edges E ($m=|E|$). Vertex $1\in V$ denotes the depot, shared by all vehicles. Set E represents all possible translations of the vehicles, moving from one vertex to another.
- $d=(d_{ij})$: the distance matrix between all pairs of vertices of V. We assume that $d_{ij}=d_{ji}$ and define $d_{ii}=0,\forall i\in V$.
- $R\geq 0$: a non-negative real valued parameter.
- $\omega(i)=\{j\in V : d_{ij}\leq R\}$, $\forall i\in V$: the set of vertices that are close enough to i, i.e., those vertices whose distance to i does not exceed R. Note that under our definition of $d_{ii}, i\in\omega(i),\forall i\in V$.

A solution to MMSVRP in G is a collection of K constrained routes. Each route $k\in\mathcal{K}$ starts in 1, spans a set $V_k\backslash\{1\}$ of selected vertices and ends at 1. We refer to the subgraph of G implied by each route k as $H_k=(V_k,E_k)$. Accordingly, $H:=\bigcup_{k=1}^{K}(V_k,E_k)$ denotes the subgraph associated to the whole set of K routes. In what follows, we say that i is covered by j if there exists $k\in\mathcal{K}$ such that $j\in V_k$ and $i\in\omega(j)$. In this case, we also say that i is covered by route k. If $j\in\bigcup_{k=1}^{K}V_k$, we say that j is visited or spanned by a route. Of course, whenever a vertex is visited by a route, it is also covered by it.

We define $w(H_k):=\sum_{\{i,j\}\in E_k}d_{ij}$ as the length of the k-th route. The cost of a feasible solution H to MMSVRP is given by $w(H)=\max\{w(H_k) : k=1,\ldots,K\}$. Through the paper, we will interchangeably use the terms routes and vehicles.

MMSVRP is clearly NP-hard, since the Traveling Salesman Problem [31,32] is one of its special cases, when $K=1$ and $\omega(i)=\{i\},\forall i\in V$.

3. Exact algorithms for MMSVRP

In this section, we present two exact algorithms to solve MMSVRP. The first one is a Branch-and-cut (BC) algorithm [33] while the second is a Local Branching (LB) [34] method that uses BC as its inner solver. Both algorithms are based on the formulation for MMSVRP presented next.

3.1. Integer programming formulation

In order to present an Integer Programming (IP) formulation for MMSVRP, assume that $\mathbb{R}$ denotes the set of real numbers and that $\mathbb{B}:=\{0,1\}$. Assume as well that, given $W\subset V$, $E[W,V\backslash W]:=\{\{i,j\}\in E : i\in W, j\in V\backslash W\}$ is the set of edges in the cut $[W,V\backslash W]$, $E[W]:=\{\{i,j\}\in E : i,j\in W\}$ is the set of edges of E with both endpoints in W, and that $\delta(i):=E[\{i\},V\backslash\{i\}]$ denotes the set of edges incident to i.

MMSVRP can be formulated in a canonical way, by using the following decision variables:

- $x_{ij}^k\in\mathbb{B}$, $\forall\{i,j\}\in E$, $\forall k\in\mathcal{K}$, assuming value 1 if edge $\{i,j\}$ belongs to the k-th route (0, otherwise);
- $y_i^k\in\mathbb{B}$, $\forall i\in V$, $\forall k\in\mathcal{K}$ assuming value 1 if i is spanned by the k-th route (0, otherwise) and,
- $w\in\mathbb{R}$, denoting the length of the longest route.

A formulation for MMSVRP can then be stated as

$$\min\{w : (w,x,y) \in \mathcal{P}_u \cap (\mathbb{R},\mathbb{B}^{Km},\mathbb{B}^{Kn})\}, \tag{1}$$

where polyhedron $\mathcal{P}_u$ is given by:

$$w \geq \sum_{\{i,j\} \in E} d_{ij}x_{ij}^k, \quad \forall k \in \mathcal{K}, \tag{2}$$

$$\sum_{\{i,j\} \in \delta(i)} x_{ij}^k = 2y_i^k, \quad \forall i \in V,\ \forall k \in \mathcal{K}, \tag{3}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} y_j^k \geq 1, \quad \forall i \in V, \tag{4}$$

$$\sum_{k \in \mathcal{K}} y_i^k \leq 1, \quad \forall i \in V\backslash\{1\}, \tag{5}$$

$$\sum_{\{i,j\} \in E[W,V\backslash W]} x_{ij}^k \geq 2y_z^k, \quad \forall W \subset V, \quad 1 \in W, \quad z \notin W, \quad \forall k \in \mathcal{K}, \tag{6}$$

$$0 \leq x_{ij}^k \leq 1, \quad \forall k \in \mathcal{K},\ \forall \{i,j\} \in E, \tag{7}$$

$$y_1^k = 1, \quad \forall k \in \mathcal{K}, \tag{8}$$

$$0 \leq y_i^k \leq 1, \quad \forall k \in \mathcal{K},\ \forall i \in V\backslash\{1\}. \tag{9}$$

Constraints (2) force w to assume a value at least as large as the length of each route and, therefore, allow us to minimize the longest of them.

Matching constraints (3) imply that whenever i is visited by the k-th vehicle, exactly two edges must be incident to i in the k-th route. Due to (8), there must be $2K$ edges incident to 1 and, therefore, K routes must be chosen.

Set covering inequalities (4) assure that each vertex is close enough to a vertex that is visited by a route, or is visited itself. Set packing constraints (5), on the other hand, impose that a vertex cannot be visited by two or more vehicles. These packing constraints are not necessary, however, if the distance matrix d satisfy the triangle inequalities.

Finally, Generalized Subtour Elimination Constraints (GSECs) (6) prevent the model from choosing routes that do not visit the depot. Note that such constraints also impose that at least three vertices will be visited in any route. Although this can be regarded as a limitation of formulation $\mathcal{P}_u$, we believe this is not a major problem because tours spanning only two vertices are unlikely to appear in optimal solutions, since the lengths of the selected K routes would be very unbalanced.

3.1.1. Breaking formulation symmetries

Due to the min–max objective function of MMSVRP, variables x and y were indexed by $k \in \mathcal{K}$, in order to state constraints (2). Consequently, if $K \geq 2$, the subgraph H of any feasible solution to MMSVRP in G may map into different points in polyhedron $\mathcal{P}_u$. For an example, assume that $K = 2$. One solution in P_u that corresponds to the subgraph of G implied by H is obtained by setting $y_i^1 = 1, \forall i \in V_1$, $y_i^2 = 1, \forall i \in V_2$, $x_{ij}^1 = 1, \forall \{i,j\} \in E_1$ and $x_{ij}^2 = 1, \forall \{i,j\} \in E_2$. Another different solution in P_u can be obtained by simply changing the route assignments: $y_i^2 = 1, \forall i \in V_1$, $y_i^1 = 1, \forall i \in V_2$, $x_{ij}^2 = 1, \forall \{i,j\} \in E_1$ and $x_{ij}^1 = 1, \forall \{i,j\} \in E_2$.

Branch-and-bound algorithms based on formulations that allow too many symmetrical solutions are likely to perform poorly. This is true since they tend to investigate identical branches within the enumeration tree (see Mendéz-Díaz and Zabala [35] and Benavent et al. [26] for other discussions on this matter). In an attempt to go around that, we investigated the inclusion of two sets of *symmetry breaking* constraints, one at a time, to the SSRLP formulation above.

The first set, given by

$$\sum_{\{i,j\} \in E} d_{ij}x_{ij}^k \geq \sum_{\{i,j\} \in E} d_{ij}x_{ij}^{k+1}, \quad \forall k \in 1,\ldots,K-1 \tag{10}$$

forces routes to be indexed in a non-increasing order of their lengths.

The idea in the second set of symmetry breaking constraints is the following. The first constraint in the set imposes that one given vertex, say i_1, must be covered by route 1. The second constraint imposes that another vertex, $i_2 : i_2 \neq i_1$, must be covered by route 1 or by route 2. Accordingly, the last constraint forces a vertex $i_{K-1} \notin \{i_1,\ldots,i_{K-2}\}$ to be covered by one of the routes in $\{1,\ldots,K-1\}$.

More precisely, given any ordered set of $K-1$ vertices of V, $\{i_1,i_2,\ldots,i_{K-1}\}$, the second set of constraints can be stated as

$$\sum_{k=1}^{t} \sum_{j \in \omega(i_t)} y_j^k \geq 1, \quad t = 1,\ldots,K-1. \tag{11}$$

Any arbitrary ordered set of vertices $\{i_1,i_2,\ldots,i_{K-1}\}$ may be chosen to define constraints (11). We found better computational results when choosing the vertices with the lowest values of $|\omega(i)|$ to be included in the set. Therefore, in our implementation, we pick the first $K-1$ vertices of V with the lowest values of $|\omega(i)|$ to define the constraints. Ties are broken arbitrarily.

One should note that symmetry breaking constraints (10) and (11) cannot be imposed simultaneously. After testing the impact of each one on BC, we concluded that better results were obtained when the second set was imposed. Therefore, constraints (11) were included in the MMSVRP formulation P_u, to avoid symmetrical solutions.

3.2. A Branch-and-cut algorithm for MMSVRP

A valid Linear Programming (LP) relaxation in our BC algorithm is implied by

$$\min\{w : (w,x,y) \in \mathcal{P}'\}, \tag{12}$$

where polyhedral region $\mathcal{P}'$ is given by the intersection of constraints (2)–(5), (7)–(9) and (11).

Assume that (12) was solved and let $(\overline{w},\overline{x}^1,\ldots,\overline{x}^K,\overline{y}^1,\ldots,\overline{y}^K)$ be its optimal solution. Denote by $\overline{G}_k = (\overline{V}_k,\overline{E}_k)$ the subgraph of G implied by $(\overline{x}^k,\overline{y}^k)$, where $\overline{V}_k = \{i \in V : \overline{y}_i^k > 0\}$ and $\overline{E}_k = \{\{i,j\} \in E : 0 < \overline{x}_{ij}^k \leq 1\}$.

If $(\overline{x}^k,\overline{y}^k)$ is integer and $\overline{G}_k$ is two-connected for all $k \in \mathcal{K}$, then $(\overline{w},\overline{x}^1,\ldots,\overline{x}^K,\overline{y}^1,\ldots,\overline{y}^K)$ solves (1). Otherwise, we attempt to strengthen the LP bound given by (12) by appending two classes of violated valid inequalities to $\mathcal{P}'$: GSECs (6) and blossom inequalities [36].

For a given k, the exact separation of GSECs can be conducted in polynomial time by a series of max-flow (min-cut) computations. More precisely, for each $i \in V\backslash\{1\}$, we seek for the minimum capacity cut that separates 1 from i, in the network defined by $\overline{G}_k$ and arc capacities $\{\overline{x}_{ij}^k : \{i,j\} \in \overline{E}_k\}$. Assume that $[W,V\backslash W]$ defines such minimum capacity cut ($1 \in W$). Whenever the capacity $\sum_{\{i,j\} \in E[W,V\backslash W]} \overline{x}_{ij}^k$ is strictly less than $2\overline{y}_z^k$ for $z \notin W$, a violated GSEC is found.

Aiming to improve the quality of the cuts being added to relaxation (12), not all violated GSECs found are included in the model. In order to decide which of them should be used to reinforce $\mathcal{P}'$, after each round $k \in \mathcal{K}$ of min-cut computations, we normalize the cuts (using the Euclidean norm) and order them according to their violation. Then, we append the most violated one in $\mathcal{P}'$. Subsequent violated inequalities, in increasing order of violation, are checked to be included in $\mathcal{P}'$. New cuts are only added if they are orthogonal enough with the set of cuts previously added to

the model. By orthogonal enough, we mean that the inner product between the candidate cut and a previously included cut should be no greater than a threshold value ε. After some experimentation, we set $\varepsilon = 0.04$.

Another set of valid inequalities for MMSVRP is the Blossom Inequalities (BIs):

$$\sum_{\{i,j\} \in E[C]} x_{ij}^k + \sum_{\{i,j\} \in T} x_{ij}^k - \sum_{i \in C} y_i^k \leq \left\lfloor \frac{|T|}{2} \right\rfloor, \quad C \subset V, \quad T \subseteq E[C, V \setminus C], \quad k \in \mathcal{K}, \tag{13}$$

where $|T|$ is odd and $|C| \geq 3$.

To separate BIs for each $k \in \mathcal{K}$, we implemented the heuristic proposed in [37]. In order to describe how the procedure works, define $\hat{E}_k := \{\{i,j\} \in E : 0 < \bar{x}_{ij}^k < 1\}$. Now assume that $\hat{G}_k = (V, \hat{E}_k)$ has p connected components, denoted by $\{\hat{G}_k^j := (\hat{V}_k^j, \hat{E}_k^j) : j = 1, \ldots, p\}$. For each value of $k \in \mathcal{K}$, we only search for violated BIs implied by candidate handles C chosen among the sets of vertices $\{\hat{V}_k^j : j = 1 \ldots, p\}$. To maximize the chances of finding a violated inequality, given handle $C = \hat{V}_k^j$, the corresponding tooth T is chosen, considering only edges in $\{\{i,j\} \in E[\hat{V}_k^j, V \setminus \hat{V}_k^j] : \bar{x}_{ij}^k = 1\}$. A procedure similar to the one described above for selecting orthogonal cuts to include in $\mathcal{P}'$ was applied.

In our approach, we keep looking for violated GSECs and BIs as long as at least one violated cut is found. Otherwise, we branch on variables, according to the strong branching approach [38], giving more priority to branch on y variables. An initial valid upper bound provided by the hybrid GRASP heuristic described in Section 4.1 is used to warm start BC.

3.3. A local branching algorithm for MMSVRP

The second exact algorithm implemented here for solving MMSVRP is a Local Branching method [34], where the Branch-and-cut algorithm described previously is used as a black box tool to explore suitable solution subspaces defined and controlled at a strategic level by a simple external branching framework. In particular, our LB algorithm uses the local branching framework implemented and tested in [39], available for download from [40].

Assuming that an initial feasible solution $H = \bigcup_{k=1}^{K} (V_k, E_k)$ for MMSVRP is available, the external branching framework is started by solving the subproblem

$$\min\{w : (w,x,y) \in \mathcal{P}_u^1 \cap (\mathbb{R}, \mathbb{B}^{Km}, \mathbb{B}^{Kn})\}, \tag{14}$$

where $\mathcal{P}_u^1$ is given by the intersection of $\mathcal{P}_u$ and the local branching constraint

$$\sum_{k \in \mathcal{K}} \sum_{i \in V_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus V_k} y_i^k \leq M. \tag{15}$$

Note that parameter M determines the size of the neighborhood of H being investigated. Therefore, by solving the first subproblem (14), we are looking for solutions that differ from H by at most M route assignments. After some experimentation, parameter M was set to 7.

After (14) is solved, other subproblems that search for feasible solutions in the domain given by the intersection of $\mathcal{P}_u \cap (\mathbb{R}, \mathbb{B}^{Km}, \mathbb{B}^{Kn})$ and

$$\sum_{k \in \mathcal{K}} \sum_{i \in V_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus V_k} y_i^k \geq M + 1$$

must be solved. Such subsequent subproblems are formulated and solved using the usual Local Branching way (see [34] for details).

Whenever a subproblem takes more than 1200 s to be solved, its resolution is halted and another subproblem, similar to the previous one, is formulated according to the diversification strategy suggested in [34]. The procedure consists in replacing parameter M by $\lfloor M/2 \rfloor$ in the local branching constraint that defines the subproblem.

To formulate the first local branching constraint (15), we also used the solution provided by the hybrid GRASP, to be described in Section 4.1.

4. Heuristic solution methods for MMSVRP

Due to the difficulty we found to solve MMSVRP to proven optimality, we introduce a heuristic approach for the problem, in the hope of obtaining solutions of better quality than those provided by BC and LB, with less computational effort.

Before presenting the proposed algorithm, let us first discuss an Integer Programming Reformulation for MMSVRP, that will be the basis for the development of the Column Generation Heuristic (CGH) introduced here.

To that aim, assume that $\mathcal{Q}$ denotes the set of all possible routes spanning the depot and at least two of the remaining vertices in $\{2, \ldots, n\}$. Given a route $q \in \mathcal{Q}$, assume that:

- $V(q)$ denotes the set of vertices visited by q;
- $E(q)$ denotes the set of edges of E used by q and $d^q := \sum_{\{i,j\} \in E(q)} d_{ij}$ represents its length;
- $\Omega(q) := \bigcup_{i \in V(q)} \omega(i)$, denotes the set of vertices visited and covered by q.

If we use variables $\lambda^q \in \mathbb{B}, \forall q \in \mathcal{Q}$ to select routes to be included in a MMSVRP solution ($\lambda^q = 1$ if q is selected, $\lambda^q = 0$, otherwise), a set covering reformulation for the problem is given by

$$\min\{w : (\lambda, w) \in \mathcal{P}_{sc} \cap (\mathbb{B}^{|\mathcal{Q}|}, \mathbb{R})\}, \tag{16}$$

where $\mathcal{P}_{sc}$ is given by:

$$w \geq \lambda^q d^q, \quad \forall q \in \mathcal{Q}, \tag{17}$$

$$\sum_{q: i \in \Omega(q)} \lambda^q \geq 1, \quad \forall i \in V, \tag{18}$$

$$\sum_{q: i \in V(q)} \lambda^q \leq 1, \quad \forall i \in V, \tag{19}$$

$$\sum_{q \in \mathcal{Q}} \lambda^q = K, \tag{20}$$

$$0 \leq \lambda^q \leq 1, \quad \forall q \in \mathcal{Q}. \tag{21}$$

Constraints (17) impose that w assumes a value at least as large as the longest route length. Inequalities (18) guarantee that each vertex will be covered by a route, while inequalities (19) enforce that a vertex is visited by at most one vehicle. Finally, constraint (20) imposes that K routes will be chosen in the model.

Assume now we are given a set of routes $\mathcal{Q}_r \subset \mathcal{Q}$ such that $|\mathcal{Q}_r| \ll |\mathcal{Q}|$. Assume as well that at least one feasible solution for MMSVRP could be found by combining K routes out of those in $\mathcal{Q}_r$. It should be clear that a valid upper bound on w could be obtained by solving the Restricted Integer Program (RIP) given by

$$\min\{w : (\lambda, w) \in \mathcal{P}_{rsc} \cap (\mathbb{B}^{|\mathcal{Q}_r|}, \mathbb{R})\}, \tag{22}$$

where $\mathcal{P}_{rsc}$ is defined by:

$$w \geq \lambda^q d^q, \quad \forall q \in \mathcal{Q}_r, \tag{23}$$

$$\sum_{q: i \in \Omega(q)} \lambda^q \geq 1, \quad \forall i \in V, \tag{24}$$

$$\sum_{q \in \mathcal{Q}_r} \lambda^q = K, \tag{25}$$

$$0 \le \lambda^q \le 1, \quad \forall q \in \mathcal{Q}_r \tag{26}$$

after applying a post-processing algorithm to its optimal solution, to remove vertices visited more than once from all but one of the K chosen routes.

Heuristic CGH explores such idea. Roughly speaking, CGH attempts to generate an attractive set $\mathcal{Q}_r$ of routes to formulate and solve model (22). After (22) is solved for a given set $\mathcal{Q}_r$, CGH attempts to use the optimal solution to (22) to build a new improved set $\mathcal{Q}_r$. In the sequence, a new model (22) is then formulated and solved for the new set of promising routes. This two-step process is repeated as long as the resolution of RIP improves on the best known solution to MMSVRP.

Our method differs from other approaches in the literature that also formulate and solve a restricted integer program to provide upper bounds for routing problems in the following sense. In the algorithms proposed in [41], for example, the set of candidate routes to be combined into a full solution to the problem, are those that are priced during the delayed column generation phases in a Branch-and-price algorithm. More precisely, from time to time, the heuristic in [41] consists in solving a restricted integer program formulated with the set of columns in the (restricted) master program in hands.

Another approach that also formulates a RIP to provide upper bounds for a routing problem is the one described by Chen and Xu [42]. In that reference, the authors proposed a column generation heuristic to tackle the Dynamic Vehicle Routing Problem with time windows. In their approach, once a RIP is formulated, only its LP relaxation is solved. This is accomplished in order to define routes (those whose decision variables are basic at the optimal solution to the LP relaxation of the restricted model) to which local searches methods will be applied.

Motivated by the fact that the constraint set in RIP has a special structure (each entry in is either 0 or 1) and the number of nonzero entries in the matrix is typically small, we attempted to solve RIP to optimality. Actually we solve various RIP, each one formulated by an improved set of candidate routes $\mathcal{Q}_r$.

Therefore, a critical issue in the approach presented here is how these sets of candidate routes $\mathcal{Q}_r$ are generated and improved, before the resolution of each RIP. In our approach, we use a hybrid GRASP [30] implementation (as well as its main ingredients like a constructive algorithm, local searches and a diversification mechanism) to initialize and update these sets. Details on how the hybrid algorithm was implemented and how its ingredients are combined into CGH are given next.

4.1. GRASP for MMSVRP

GRASP (Greedy Randomized Adaptive Search Procedure) is a class of stochastic search algorithms that uses randomized greedy constructive search heuristics to generate a large number of different candidate solutions to optimization problems.

At each GRASP iteration, the solution obtained with the randomized constructive phase is submitted to a local search procedure. This two-phase process is iterated until a termination criterion is satisfied; usually, after a maximum number of iterations are performed or a time limit is reached. We call hybrid our implementation of GRASP because the usual local search step is replaced by an Iterated Local Search Method (ILS) [43].

4.1.1. Constructive heuristic

The constructive heuristic in GRASP for MMSVRP is based on the Cheapest Insertion Algorithm (CIA) [44] proposed for the Euclidean Traveling Salesman Problem, which can be easily adapted to MMSVRP, resulting in algorithm CIA_MMSVRP. The main idea in CIA_MMSVRP is to simultaneously construct K tours covering the n vertices, in an iterative process that adds one vertex at a time to the shortest of the K routes (initially started with the depot and one randomly chosen vertex) until all vertices are covered.

For the TSP, the selection policy that chooses a vertex to be included in the partial solution is based on the cheapest insertion rule. In practice, we observed that, for the MMSVRP case, the cheapest insertion rule may not be the best selection policy. When deciding which vertex should be included in a given route, one has to balance two factors: the cost of expanding the route and the number of vertices that still will remain uncovered after the expansion takes place. Therefore, in our implementation of CIA_MMSVRP, the vertex that expands the route is the one that minimizes $\Delta := \{\Delta^j - \delta|\overline{\omega}(j)| : j \text{ is not covered yet}\}$, where Δ^j is the minimum cost of including vertex j in the route in hands, $\delta \ge 0$ is a weighting factor that depends on the instance and $\overline{\omega}(j)$ is the set of sensor nodes in $\omega(j)$ that are not covered by the current partial solution. Since all instances considered in this study are Euclidean instances in the plane, δ was set to 0.15 times the largest distance between two vertices.

Let us now discuss how CIA_MMSVRP was randomized in order to be used in GRASP. Instead of choosing the vertex for which Δ is attained to add to the route, in the randomized version of CIA_MMSVRP, one randomly chooses any vertex j yet uncovered whose corresponding value of $\Delta^j - \delta|\overline{\omega}(j)|$ lies in the interval $[\Delta,(1+\alpha)\Delta]$, where parameter $\alpha \ge 0$ controls the level of randomization of the procedure. Actually, since α is not kept constant during all GRASP iterations, our procedure is a reactive GRASP (see [45] for details). In our implementation, at the first GRASP iteration, α is randomly chosen from the set $\{0.05,0.10,\ldots,0.50\}$ with uniform probability. As the algorithm evolves, higher probabilities are assigned to those values of α that result in higher quality solutions.

4.1.2. Local searches and diversification mechanism

Two local search (LS) procedures, 2-OPT [46] and 2-SWAP [47], and a diversification mechanism, named the Nodes Reinsertion Algorithm (NRA) [48], were embedded in hybrid GRASP.

At any iteration of 2-OPT, the search is always applied to the longest route in the feasible solution being investigated. It works by removing two edges from the longest route and then trying to reconnect the resulting two paths using less expensive edges. If the length of the route becomes smaller than the length of another route in the solution, the search switches to the new longest route. 2-OPT implemented here makes use of the do not look bits neighborhood reduction [49] to cut down CPU running times.

In 2-SWAP, we try to improve the solution by swapping nodes from their routes. Assuming that two vertices $p \in V_{k_1}$ and $q \in V_{k_2}$ are given, one replaces p by q and q by p. Note that after swapping the two vertices, only two routes change: V_{k_1} becomes $(V_{k_1}\backslash\{p\}) \cup \{q\}$ and V_{k_2} becomes $(V_{k_2}\backslash\{q\}) \cup \{p\}$. In case $k_1 = k_2$, only the visiting order of p and q is changed.

One weakness of these two neighborhoods is that the whole set of visited vertices is preserved after their application. To overcome that, we also make use of NRA, which, in order to explore a broader solution space, combines two phases. In the first one, each visited vertex is attempted to be removed from its route, according to a given probability. Whenever a vertex $p \in V_k$ (together with its incident edges) is removed from route k, the two neighbors of p in the route are joined by an edge, in order to establish a Hamiltonian circuit spanning the remaining set of vertices $V_k\backslash\{p\}$. Since the set of routes obtained after the last vertex removal is not necessarily feasible (there may exist uncovered vertices), in the second phase, CIA_MMSVRP is applied to recover feasibility. In doing so, different

vertices from those that were removed in the preceding phase may possibly be added, generating a new feasible solution.

4.1.3. The hybridization of GRASP

The hybridization of GRASP takes place by replacing its usual local branch step by an Iterated Local Search Method (ILS) [43].

ILS is a randomized algorithm that, first, applies a local search procedure to a feasible solution for the problem being solved. In the sequence, the local optimum is submitted to a perturbation procedure, originating a new intermediate solution. Then, the local search step takes place once again: local search procedures are applied to this intermediate solution, obtaining a (possibly) new local optimum. This process of alternating local searches and perturbation steps goes on until a satisfaction criterion is met.

The ILS implemented here is a hybrid procedure itself since we replaced its internal local search phase by a Variable Neighborhood Descent (VND) [50]. Roughly speaking, VND is a local search method based on the idea of iteratively switching neighborhoods during the course of the search. The neighborhood structures are ordered according to their complexity: the investigation of cheaper neighborhoods precede the investigation of more expensive neighborhoods. Accordingly, one first performs a local search algorithm exploring the cheapest neighborhood. Once a local minimum is found, a local search algorithm based on the next neighborhood structure starts. In this process, whenever an improved solution is found, the search restarts from the cheapest neighborhood. The procedure goes on, until the best solution in hands is found to be a local minimum with respect to all neighborhoods. In our implementation, the VND is composed of methods 2-OPT and 2-SWAP, to be scanned in that order. The perturbation step in ILS is NRA.

4.2. Integrating hybrid GRASP into a column generation heuristic for MMSVRP

CGH is a heuristic for MMSVRP in which we use hybrid GRASP and its main ingredients to create a candidate set of routes $\mathcal{Q}_r$ to formulate (22), which can be solved by a Linear Programming based Branch-and-bound algorithm. In the sequence, we use the solution to the RIP to create a new set $\mathcal{Q}_r$. Another RIP, formulated for the new set $\mathcal{Q}_r$, is then solved. The process is repeated, as long as an improving solution is found in each round. The procedure therefore works in an iterative way, ruled according to the following steps:

(1) (populate) Apply a certain number of hybrid GRASP iterations. Store each set of K routes associated to the best solution found in each GRASP iteration in set $\mathcal{Q}_1$. Keep record of the best solution found during all GRASP iterations. The number of GRASP iterations is chosen such that $|\mathcal{Q}_1| \geq 12\,500$.
(2) (recombine) Apply the following steps, as many times as needed, in order to obtain set $\mathcal{Q}_2$ of the desired size:
  (a) Let k_1 be an integer randomly chosen from $\{1,\ldots,K-1\}$.
  (b) Randomly choose k_1 routes out of those in $\mathcal{Q}_1$.
  (c) Apply CIA_MMSVRP to the partial solution given by these k_1 routes, obtaining a feasible solution to MMSVRP by constructing additional $K-k_1$ routes.
  (d) Store the newly constructed routes in $\mathcal{Q}_2$.
(3) (post-processing) Given the best solution H^* to MMSVRP, remove from $\mathcal{Q}_1 \cup \mathcal{Q}_2$ all those routes whose length is greater than $w(H^*)$. Let $\mathcal{Q}_r$ be the remaining set of routes. Set $\mathcal{Q}_1 \leftarrow \emptyset$ and $\mathcal{Q}_2 \leftarrow \emptyset$.
(4) (RIP) Solve (22).
(5) Remove vertices visited more than once from all but one route in the solution to (22) and apply VND to it.
(6) If this solution does not improve the best overall solution (prior to the resolution of (22)), stop. Otherwise:
  (a) (re-populate) Apply ILS to the best solution found so far, as many times as needed, storing all the routes generated in set $\mathcal{Q}_1$, until $|\mathcal{Q}_1| \geq 5000$.
  (b) Go to the recombination step above and repeat the process.

As it could be appreciated from the schematic above, hybrid GRASP is used to first populate set $\mathcal{Q}_1$. In the recombination phase that follows, we use set $\mathcal{Q}_1$ to generate new routes. This is accomplished by trying to build new solutions from partial solutions that combine k_1 routes in $\mathcal{Q}_1$. When ILS is applied to the solution to (22) in step 6-a, we take advantage of the local searches and of NRA to build a new diversified set of promising routes $\mathcal{Q}_1$, to restart the algorithm.

In the first time the recombination step is called, we impose that $\mathcal{Q}_2$ has at least 35 000 routes. In the subsequent calls, we impose that $\mathcal{Q}_2$ has at least 15 000 columns. The desired sizes of sets $\mathcal{Q}_1$, $\mathcal{Q}_2$ (and therefore of $\mathcal{Q}_r$) were chosen after some experimentation, trying to balance the solution quality and the computational effort required to solve (22).

5. Computational experiments

In this Section, computational experiments involving BC, LB and CGH are reported. Our experiments were conducted with 20 test instances generated from two dimensional Euclidean TSPLIB [51] instances. For each TSPLIB instance considered here, a corresponding MMSVRP instance was generated by defining parameter R such that $\sum_{i \in V} |\omega(i) \setminus \{i\}|/n$ was as close to 0.75 as possible. This seems to be a good value for sparse networks, if we consider the application in WSN that MMSVRP relates to. Distances d_{ij} were obtained by rounding the corresponding Euclidean distances to the nearest integer. Each MMSVRP instance was tested for $K \in \{2,3,4\}$.

Our implementation of BC makes use of CPLEX [52] (release 10.2) callback routines to manage the enumeration tree and cut generation. In BC, branching on variables was performed using pre-implemented CPLEX strong branching options. All CPLEX built in procedures for pre-processing and heuristics for finding feasible integer solutions were turned off. CPLEX solver is also used to solve each RIP (22) in the course of CGH.

All algorithms discussed here were implemented in `C++` and experiments were conducted on a Intel Core 2 Duo machine with 3.0 Ghz and 4 Gbytes of RAM memory, under Linux Operating System. `gcc` compiler was used with `-O3` optimization flags turned on.

5.1. Computational results with the exact solution methods

In order to fairly compare BC and LB, the same feasible solution to MMSVRP (provided by the GRASP discussed in Section 4.1) was used in the initialization of both methods. A time limit of 4 h was imposed on the execution times of both approaches, for each instance and value of K.

For almost all cases, neither BC nor LB managed to solve the instances within the imposed time limit. Therefore, for BC, we report the best lower and upper bounds on w, when the time limit was reached. Usually, when LB does not solve the problem within the time limit, the external enumeration tree is very unbalanced and thus, the lower bounds obtained by LB tend to be poor. Because of that, lower bounds provided by LB are not reported here. Since no other lower (nor upper) bounding procedures are available in the MMSVRP literature, we only compared BC lower bounds to those bounds implied by a scheme suggested by Glaab [25].

Glaab [25] noticed that a valid lower bound on the shortest longest route in a VRP problem could be obtained by any relaxation

that replaces the original min–max objective function by the average route length. Based on such fact, Glaab proposed combinatorial relaxations for the VRP under investigation in that reference.

Therefore, in an attempt to eventually obtain shaper lower bounds to MMSVRP, we tried to evaluate the bound

$$w_K = \frac{1}{K}\ \min\left\{ \sum_{\{i,j\}\in E} d_{ij}x_{ij} : (x,y)\in \mathcal{P}_K \cap (\mathbb{B}^m,\mathbb{B}^n) \right\}, \tag{27}$$

where formulation $\mathcal{P}_K$ is obtained after dropping index k from variables x^k,y^k, eliminating constraints (2) and then redefining formulation $\mathcal{P}_u$ accordingly.

In order to solve (27), we used the same implementation of the branch-and-cut algorithm described in Section 3.2, under the same time limit of 4 h. If the time limit is reached and (27) is left unsolved, the best lower bound $\underline{w}_K$ on w_K (which is also a lower bound on w) is retrieved.

In Tables 1–3, we present the main computational results attained by BC and LB for each instance in our test bed, for $K=2,3$ and 4, respectively. In the first and second columns of the tables, we indicate the instance name followed by its id (from 1 to 20). The next two entries are the lower bound $\underline{w}_K$ and the initial upper bound given by GRASP. In the next four columns, we present results for BC. They are: the best upper (ub) and lower (lb) bounds obtained when the time limit was reached, the implied duality gap, $100(ub-lb)/ub$, and the time ($t(s)$) in seconds taken by BC to find the corresponding best upper bound. Similar entries for LB, ub and $t(s)$, are given in the subsequent two columns. The computing times reported for LB in the last column of Tables 1–3 refer to the total computing times involved in LB, including the times need by the inner BC algorithm to solve each subproblem.

An entry "−" in columns under headings $t(s)$ indicates that the corresponding algorithm was not able to improve the initial upper bound, within the imposed time limit.

Table 1
Branch-and-cut and local branching results, $K=2$.

Instance		$\underline{w}_K$	GRASP	Branch-and-cut				Local branching	
Name	Id			ub	lb	Gap (%)	$t(s)$	ub	$t(s)$
eil51	1	187.0	196	193	193.0	–	78	193	31
st70	2	325.0	342	342	342.0	–	–	342	–
eil76	3	238.0	261	248	241.2	2.8	7166	249	1562
rat99	4	520.2	610	582	536.5	7.8	6387	576	7031
KroA100	5	9877.0	10577	10350	10246.0	1.0	1965	10352	2835
eil101	6	305.0	327	309	306.9	0.7	4660	313	3008
lin105	7	6946.0	8153	8015	7883.6	1.6	13148	8074	6819
pr107	8	17302.0	26534	26534	23175.3	12.7	–	26425	4489
pr124	9	27193.9	33912	33822	29778.5	12.0	3513	33762	4905
bier127	10	55721.4	60229	57513	56148.1	2.4	10444	56732	12112
ch130	11	2695.8	2999	2915	2677.7	8.14	11598	2869	10506
pr136	12	30721.6	43520	43458	33042.4	24.0	6040	40251	8093
pr144	13	24279.6	34139	34139	26511.8	22.3	–	34065	4191
ch150	14	2535.5	3089	3089	2568.0	16.9	–	2953	14126
kroA150	15	11160.7	12400	12251	10975.2	10.4	10729	12379	2451
pr152	16	34658.7	43202	42654	39962.98	6.31	11569	42983	6940
u159	17	17994.0	21378	21378	18763.9	12.23	–	21040	2173
rat195	18	705.5	993	993	718.2	27.63	–	979	2508
kroA200	19	10836.6	13480	13480	11233.0	16.7	–	13479	2438
tsp225	20	1348.9	1780	1780	1383.1	22.3	–	1780	–

Table 2
Branch-and-cut and local branching results, $K=3$.

Instance		$\underline{w}_K$	GRASP	Branch-and-cut				Local branching	
Name	Id			ub	lb	Gap (%)	$t(s)$	ub	$t(s)$
eil51	1	130.0	143	141	141.0	–	455	141	45
st70	2	224.0	272	270	262.7	2.7	9150	270	142
eil76	3	161.1	188	178	166.7	6.4	1060	181	3147
rat99	4	354.8	507	495	394.8	20.2	7373	506	2406
KroA100	5	6738.0	7623	7499	7146.1	4.71	6331	7512	3089
eil101	6	208.0	238	238	207.5	12.8	–	230	2988
lin105	7	4770.7	7050	7040	5274.8	25.1	13251	7027	2139
pr107	8	11844.0	22247	22247	20821.0	7.2	–	22247	–
pr124	9	18232.3	26448	26102	20065.7	23.1	12804	26448	–
bier127	10	37272.4	44082	44082	37329.3	15.3	–	43935	2416
ch130	11	1809.5	2240	2219	1799.1	18.9	10731	2238	2414
pr136	12	21027.0	33209	33209	22904.1	31.0	–	33209	–
pr144	13	16094.21	28982	28982	18306.6	36.8	–	28954	2552
ch150	14	1690.1	2331	2331	1787.4	23.3	–	2297	2435
kroA150	15	7320.70	8975	8975	7373.1	17.9	–	8930	2452
pr152	16	23328.1	38495	38495	24073.9	37.46	–	38480	2907
u159	17	12450.7	16715	16715	12949.0	22.53	–	16715	–
rat195	18	476.1	778	778	487.0	37.4	–	777	2494
kroA200	19	7207.2	9913	9913	7434.4	25.0	–	9890	2485
tsp225	20	919.6	1391	1391	933.0	32.93	–	1379	2688

Table 3
Branch-and-cut and local branching results, $K=4$.

Instance		$\underline{w}_K$	GRASP	Branch-and-cut				Local branching	
Name	Id			*ub*	*lb*	*Gap* (%)	*t*(*s*)	*ub*	*t*(*s*)
eil51	1	102.0	121	119	117.29	1.5	13537	119	99
st70	2	176.0	232	232	196.0	15.6	–	231	3199
eil76	3	127.0	153	153	126.53	17.3	–	153	–
rat99	4	276.5	464	464	276.8	40.3	–	462	2516
KroA100	5	5238.5	6596	6596	5487.6	16.80	–	6582	2409
eil101	6	159.5	185	185	160.5	13.28	–	184	2408
lin105	7	3751.3	6503	6503	3856.8	40.1	–	6503	–
pr107	8	9132.9	21125	21125	20821.0	1.4	–	21125	–
pr124	9	14144.8	23105	23105	14492.89	37.27	–	23105	–
bier127	10	28162.2	34567	34567	28324.1	18.06	–	34397	2433
ch130	11	1377.7	1762	1762	1380.01	21.63	–	1762	–
pr136	12	16368.4	28486	28486	17390.8	38.9	–	28486	–
pr144	13	12364.7	26055	26055	13724.1	47.3	–	26055	–
ch150	14	1288.5	1884	1884	1400.9	25.7	–	1884	–
kroA150	15	5584.9	7349	7349	5757.7	21.6	–	7267	2503
pr152	16	18215.0	33964	33964	19493.3	42.6	–	33964	–
u159	17	9722.8	14688	14688	10080.6	31.4	–	14688	–
rat195	18	368.9	669	669	379.5	43.3	–	668	2608
kroA200	19	5516.4	8131	8131	5606.4	31.1	–	8130	2563
tsp225	20	712.2	1196	1196	726.5	39.3	–	1196	–

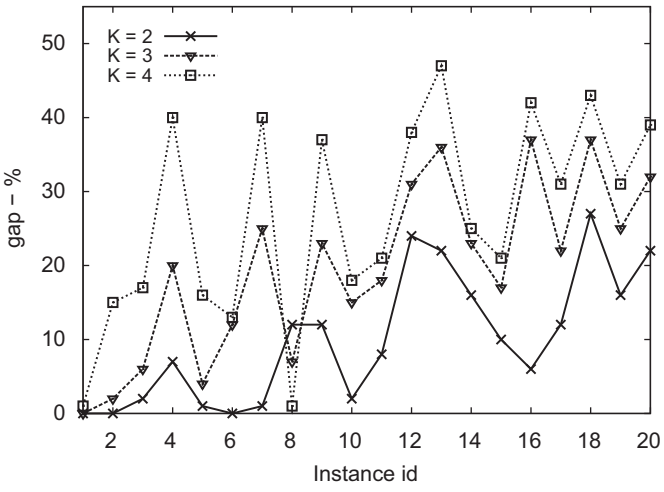

Fig. 1. Branch-and-cut duality gaps as a function of *K*.

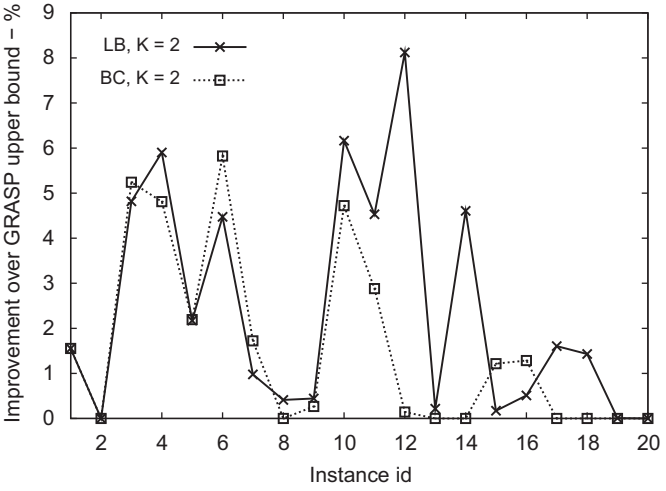

Fig. 2. Local branching and branch-and-cut improvements over initial upper bounds—$K=2$.

Table 4
Comparison of local branching and branch-and-cut upper bounds.

K	Number of improvements			Avg improvement (%)			Avg time to improve (s)		
	2	3	4	2	3	4	2	3	4
BC	12	9	1	2.55	1.79	1.65	7275	6907	13537
LB	18	16	9	3.15	1.13	1.65	5524	1912	99

The results in the tables indicate that MMSVRP is indeed very difficult to solve. BC and LB only solved instances `eil51`, `st70` with $K=2$ and `eil51` with $K=3$. On the average, the duality gaps given by BC when the time limit was achieved were 10.4%, 20.0% and 27.4%, respectively for $K=2,3,4$. In Fig. 1, we plot the BC duality gap for each instance and value of *K* tested here. Note that for almost all instances (except for `pr107`), the duality gaps grow as *K* grows. The difficulty to solve the problem thus increases significantly with the growth of *K*.

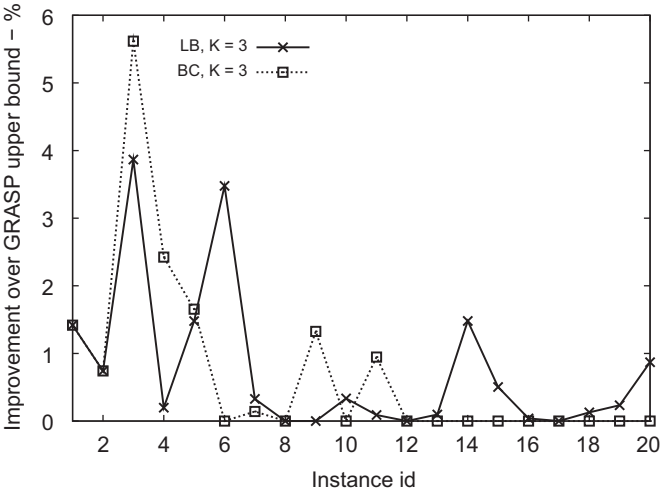

Fig. 3. Local branching and branch-and-cut improvements over initial upper bounds—$K=3$.

In five cases, the bound $\underline{w}_K$ was stronger than *lb* counterparts. Being diminute, these lower bound improvements do not significantly reduce the duality gaps still observed for MMSVRP.

In Table 4 and in Figs. 2–4, we compare BC and LB, in terms of the quality of their upper bounds. In the table, for each value of *K*, we report on the number of times each algorithm improved on the initial upper bound provided by GRASP. We also present the average improvement over the initial upper bound and the time taken to obtain that improvement, for the set of instances where both algorithms improved on the GRASP initial solution. For each value of *K*, we depict in Figs. 2–4, the improvement of the best upper bound provided by LB and BC over the initial bound given by GRASP.

Results in Table 4 and in Figs. 2–4 indicate that LB improved the initial solution in many more cases (43 for LB against 22 for BC). The advantage of LB over BC, in terms of the quality of their upper bounds, becomes more evident when higher values of *K* are considered. Another important aspect that must be pointed out is that, compared to BC, LB usually needs much less computing time to achieve a comparable upper bound improvement.

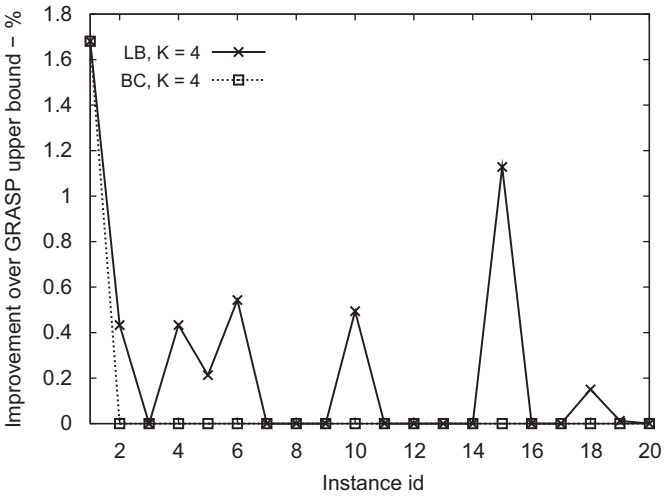

Fig. 4. Local branching and branch-and-cut improvements over initial upper bounds—K=4.

5.2. Computational results with the heuristics

In order to evaluate its robustness, CGH was executed for 10 independent trials, for each instance and value of *K*. In each trial, an independent execution of GRASP was performed. The number of GRASP iterations was defined such that each GRASP execution takes 120 s. Each of these GRASP executions originate (possibly) different initial sets of routes $\mathcal{Q}_1$ in step 1 of CGH. No time limit was imposed on the resolution of RIP (22).

In Tables 5–7, we compare the best upper bounds obtained by BC and LB to those obtained by the heuristics. The best BC and LB upper bounds are reported, once gain, in the third and in the fourth columns of the tables. In the next four columns, we present the worst and the best upper bounds obtained in the 10 executions of GRASP and of CGH. For some instances, the initial GRASP upper bound used to warm-start BC (and LB) given in Tables 1–3 lies outside the range given by the best and worst GRASP upper bounds indicated in Tables 5–7. This happens because the 10 independent GRASP executions considered here do not include the GRASP execution used to warm-start BC (and LB). In the next column (under headings $\leq$), we indicate the number of executions (out of 10) for which CGH provided upper bounds at least as good as those given by LB and BC. In the sequence, we present the average computing time (in seconds) CGH needed to run (GRASP execution times included). Finally, in the last column of the table, we indicate the duality gap, computed considering the best known upper bounds for MMSVRP and the lower bounds provided by BC. In Figs. 5 and 6, we respectively depict the improvements of the worst and the best solution found by CGH over the best solution given by BC and LB.

As it can be appreciated from Fig. 5, the worst solution found by CGH compares favorably with the best solution found by BC and LB as *n* and *K* grows. Although for instances with small values of *n* and *K* the upper bounds provided by LB and BC are usually sharper than the worst bounds given by CGH, the heuristic significantly improves the bounds for the larger instances in the test bed. In particular, results in the tables indicate that for 9,16 and 17 cases,

Table 5
CGH results, K=2.

Instance		BC	LB	GRASP		CGH				Gap (%)
Name	Id			Worst	Best	Worst	Best	*t*(*s*)	≤	
eil51	1	193	193	196	194	196	194	139	0	0
st70	2	342	342	343	342	342	342	241	10	0
eil76	3	248	249	259	255	258	250	161	0	2.4
rat99	4	582	576	608	595	586	575	188	2	6.6
kroA100	5	10350	10352	10614	10498	10350	10350	385	10	1.0
eil101	6	309	313	327	323	320	311	236	0	0.6
lin105	7	8015	8074	8275	8107	8041	8022	269	0	1.6
pr107	8	26534	26425	26544	26466	26534	26425	277	8	12.3
pr124	9	33822	33762	33912	33361	33361	33186	350	10	10.3
bier127	10	57513	56732	61257	59587	58723	57698	320	0	1.0
ch130	11	2915	2869	3005	2943	2906	2879	216	0	6.7
pr136	12	43458	40251	42990	42647	41741	39912	295	5	17.2
pr144	13	34139	34065	34158	33862	33593	33396	348	10	20.6
ch150	14	3089	2953	3072	2988	2948	2865	278	10	10.4
kroA150	15	12251	12379	12716	12400	12173	11980	579	10	8.4
pr152	16	42654	42983	44021	43202	42578	42357	463	10	5.7
u159	17	21378	21040	21658	21239	20795	20641	441	10	9.1
rat195	18	993	979	1001	971	890	870	375	10	17.4
kroA200	19	13480	13479	13577	13367	12712	12465	1147	10	9.9
tsp225	20	1780	1780	1810	1777	1691	1656	590	10	16.4
Avg										7.9

Table 6
CGH results, $K=3$.

Instance		BC	LB	GRASP		CGH				Gap (%)
Name	Id			Worst	Best	Worst	Best	$t(s)$	$\leq$	
ei151	1	141	141	143	143	143	143	138	0	0
st70	2	270	270	275	272	271	269	178	8	2.2
ei176	3	178	181	184	179	181	176	158	5	5.1
rat99	4	495	506	495	482	481	472	183	10	16.3
kroA100	5	7499	7512	7713	7510	7426	7426	161	10	3.8
ei1101	6	238	230	231	226	221	218	211	10	4.6
lin105	7	7040	7027	7060	6929	6922	6852	206	10	23.0
pr107	8	22247	22247	22247	22387	22427	22313	142	10	6.4
pr124	9	26102	26448	26577	25760	25600	22550	251	10	11.0
bier127	10	44082	43935	44072	42570	41564	40166	347	10	7.1
ch130	11	2219	2238	2188	2150	2087	2019	238	10	10.8
pr136	12	33209	33209	32275	31771	30659	30107	325	10	23.9
pr144	13	28982	28954	28283	27875	27535	27466	261	10	33.3
ch150	14	2331	2297	2254	2190	2091	2081	286	10	14.1
kroA150	15	8975	8930	9055	8765	8519	8199	248	10	10.1
pr152	16	38495	38480	38638	38041	36710	36116	370	10	33.3
u159	17	16715	16715	16795	16276	15931	15693	297	10	17.5
rat195	18	778	777	775	758	717	694	424	10	29.8
kroA200	19	9913	9890	10093	9803	9379	9107	384	10	16.2
tsp225	20	1391	1379	1394	1365	1301	1244	700	10	25.0
Avg										14.7

Table 7
CGH results, $K=4$.

Instance		BC	LB	GRASP		CGH				Gap (%)
Name	Id			Worst	Best	Worst	Best	$t(s)$	$\leq$	
ei151	1	119	119	122	120	121	119	161	4	0.8
st70	2	232	231	233	231	232	231	183	8	15.2
ei176	3	153	153	154	153	150	150	182	10	15.3
rat99	4	464	462	464	456	458	455	209	10	39.1
kroA100	5	6596	6582	6617	6391	6424	6251	212	10	12.2
ei1101	6	185	184	185	180	173	173	288	10	6.9
lin105	7	6503	6503	6535	6485	6447	6447	223	10	40.2
pr107	8	21125	21125	21125	21125	21125	21125	156	10	1.4
pr124	9	23105	23105	23337	22983	22958	22809	312	10	36.5
bier127	10	34567	34397	34938	34096	33057	31871	608	10	11.1
ch130	11	1762	1762	1779	1736	1684	1647	503	10	16.2
pr136	12	28486	28486	28595	27809	27027	26955	451	10	29.0
pr144	13	26055	26055	26121	25946	25919	25618	336	10	46.4
ch150	14	1884	1884	1905	1884	1778	1778	372	10	21.2
kroA150	15	7349	7267	7508	7265	7093	6894	397	10	16.5
pr152	16	33964	33964	34239	33964	33833	33678	236	10	42.1
u159	17	14688	14688	14708	14405	14313	14271	188	10	29.4
rat195	18	669	668	678	669	646	646	318	10	41.2
kroA200	19	8131	8130	8271	8090	7455	7455	529	10	24.8
tsp225	20	1196	1196	1213	1184	1155	1099	1181	10	33.8
Avg										26.3

the worst upper bound provided by CGH improves on the best upper bound given by LB and BC, respectively for $K = 2,3,4$. Compared to BC and LB, CGH works better exactly for those instances that BC and LB were not capable of improving the initial upper bounds. For example, in almost all executions for $K = 3,4$ (see columns under headings $\leq$ in Tables 5–7) CGH provided solutions at least as good as the best obtained by BC and LB. Not only the quality of the solutions provided by CGH is significantly better, but CGH also obtained such solutions with much less computational time. More precisely, CGH took around one tenth of the time needed by BC and LB to provide their best upper bounds.

Our results suggest that the quality of the solutions provided by CGH does not vary too much. The maximum deviation between the worst and the best upper bounds provided by CGH were 4.6%, 4.6% and 5.1% for $K=2,3$ and 4, respectively. Despite the improvements on the upper bounds, the duality gaps (considering the best upper bounds obtained in this study) we report on Tables 5–7 are still very high for many instances.

6. Conclusions

In this paper, we discussed a vehicle routing problem, named the Min-Max Selective Vehicle Routing Problem (MMSVRP), that models an application in Wireless Sensor Networks. MMSVRP differs from other routing problems in the literature because the

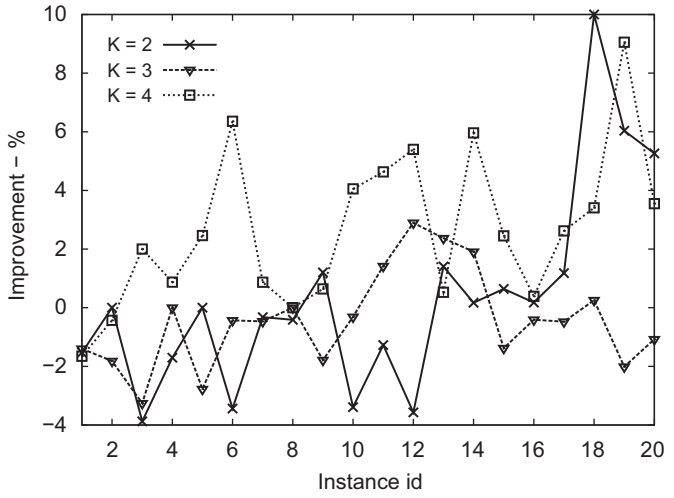

Fig. 5. Improvement of the worst solution found by CGH over the best solution given by BC and LB.

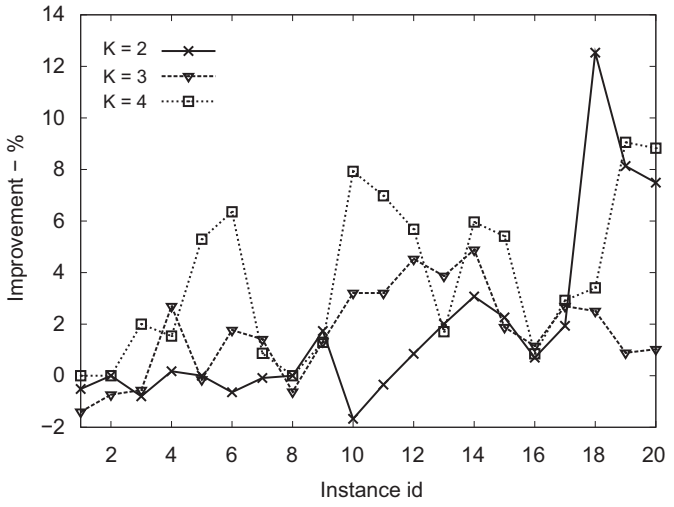

Fig. 6. Improvement of the best solution found by CGH over the best solution given by BC and LB.

goal is to minimize the longest route and not all clients need to be visited by a vehicle. Differently, clients need only to be close enough to another client that is visited by a vehicle.

We implemented two exact algorithms for solving the problem: a Branch-and-cut method and a Local Branching algorithm. Our computational experience with these algorithms indicate that the combination of the min–max objective function and the selective nature makes the problem very difficult to solve; few instances could be solved to proven optimality and huge duality gaps were obtained for most of them. According to our findings, the hardness of the problem increases as the number of vehicles grows. This happens because the Linear Programming lower bounds used to prune the Branch and bound trees become weaker, as a combination of the min–max nature of the objective function with large number of vehicles.

Motivated by the difficulties we found to solve the problem to optimality, we proposed a Column Generation Heuristic, in an attempt to provide sharper upper bounds with less computational effort. The heuristic indeed improved on the best feasible solutions provided by the exact methods, mainly for the hardest instances in our test bed, those with larger number of clients and vehicles.

Despite the improvements provided by the Column Generation Heuristic, the duality gaps we still report for MMSVRP are very high for many instances. This suggest that other lower bounding approaches should be investigated. In particular, we plan to implement a Branch-and-price algorithm based on a set covering reformulation for MMSVRP.

Acknowledgments

The authors wish to thank two anonymous referees for providing suggestions that improved the presentation of this paper.

References

[1] Fukasawa R, Longo H, Lysgaard J, de Aragão MP, Reis M, Uchoa E, et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 2006;106(3):491–511.
[2] Baldacci R, Christofides N, Mingozzi A. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 2008;115:351–85.
[3] Dantzig GB, Ramser RH. The truck dispatching problem. Management Science 1959;6:80–91.
[4] Clarke G, Wright JW. Scheduling of vehicles from a depot to a number of delivery points. Operations Research 1964;12:568–81.
[5] Christofides N, Mingozzi A, Toth P. Exact algorithms for the vehicle routing problem based on the spanning tree and shortest path relaxations. Mathematical Programming 1981;20:255–82.
[6] Fisher ML. Optimal solution of vehicle routing problems using minimum *K*-trees. Operations Research 1994;42:626–42.
[7] Martinhon C, Lucena A, Maculan N. Stronger *K*-tree relaxations for the vehicle routing problem. European Journal of Operational Research 2004;158:56–71.
[8] Baldacci R, Hadjiconstantinou E, Mingozzi A. An exact algorithm for the capacitated vehicle routing problem based on two-commodity network flow formulation. Operations Research 2004;52(4):723–38.
[9] Alvarenga GB, Mateus GR, de Tomi G. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Computers and Operations Research 2007;34(6):1561–84.
[10] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[11] Fischetti M, Toth P, Vigo D. A Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. Operations Research 1994;42:846–59.
[12] Gendreau M, Laporte G, Potvin JY. Vehicle routing: modern heuristics. In: Aarts EHL, Lenstra JK, editors. Local search in combinatorial optimization, 1997. p. 311–36.
[13] Golden BL, Wasil EA, Kelly JP, Chao IM. Metaheuristics in vehicle routing. In: Crainic TG, Laporte G, editors. Fleet management and logistics. Kluwer; 1998. p. 33–56.
[14] Toth P, Vigo D. The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications; 2002.
[15] Golden BL, Raghavan S, Wasil EA. The vehicle routing problem: latest advances and new challenges. Springer; 2008.
[16] Laporte G. Fifty years of vehicle routing. Transportation Science 2009;43(9):408–16.
[17] Laporte G, Desrochers M, Nobert Y. Two exact algorithms for the distance constrained vehicle routing problem. Networks 1984;14:161–72.
[18] Laporte G, Nobert Y, Desrochers M. Optimal routing under capacity and distance restrictions. Operations Research 1985;33:1050–73.
[19] Pullen H, Webb M. A computer application to a transport scheduling problem. Computer Journal 1967;10:10–3.
[20] Knight K, Hofer J. Vehicle scheduling with timed and connected calls. Operational Research Quarterly 1968;19:337–60.
[21] Savelsbergh MWP. Local Search in routing problems with time windows. Annals of Operations Research 1985;4:285–305.
[22] Savelsbergh MWP, Sol M. The general pickup and delivery problem. Transportation Science 1995;29:17–29.
[23] Golden BL, Baker E, Alfaro J, Schaffer J. The vehicle routing problem with backhauls: two approaches. In: Hammesfahr R, editor. Proceedings of the XXI annual meeting of S.E. TIMS; 1985. p. 90–2.
[24] Valle CA, da Cunha AS, Aioffi WM, Mateus GR. Algorithms for improving the quality of service in wireless sensor networks with multiple mobile sinks. In: Proceedings of the 11th ACM international symposium on modeling analysis and simulation of wireless and mobile systems. Vancouver, BC, Canada: ACM Press; 2008. p. 239–43.
[25] Glaab H. A new variant of a vehicle routing problem: lower and upper bounds. European Journal of Operational Research 2002;139:557–77.
[26] Benavent E, Coberán A, Plana I, Sanchis JM. Min–max K-vehicles windy rural postman problem. Networks 2009;54:216–26.
[27] Baldacci R, Dell'Amico M, Gonzalez JS. The capacitated m-ring star problem. Operations Research 2007;55(6):1147–62.

[28] Hoshino EA, de Souza CC. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. In: LAGOS'09—V Latin-American algorithms, graphs and optimization symposium. Electronic notes in discrete mathematics, vol. 35; 2009. p. 103–8.
[29] Valle CA, da Cunha AS, Mateus GR, Martinez LC. Exact algorithms for a selective vehicle routing problem where the longest route is minimized. In: LAGOS'09—V Latin-American algorithms, graphs and optimization symposium. Electronic notes in discrete mathematics, vol. 35; 2009. p. 133–8.
[30] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;6:109–33.
[31] Dantzig GB, Fulkerson DR, Johnson SM. Solution of a large scale traveling salesman problem. Operations Research 1954;2:393–410.
[32] Junger M, Reinelt G, Rinaldi G. The traveling salesman problem. In: Ball MO, editor. Handbooks in OR and MS, vol. 7. Elsevier Science Publishers; 1995. p. 225–330.
[33] Padberg MW, Rinaldi G. A branch-and-cut algorithm for resolution of large scale of symmetric traveling salesman problem. SIAM Review 1991;33:60–100.
[34] Fischetti M, Lodi A. Local branching. Mathematical Programming 2003;98: 23–47.
[35] Méndez-Díaz I, Zabala P. A Branch-and-cut algorithm for graph coloring. Discrete Applied Mathematics 2006;154:826–47.
[36] Gendreau M, Laporte G, Semet F. The covering tour problem. Operations Research 1997;45(4):568–76.
[37] Naddef D, Thienel S. Efficient separation routines for the symmetric traveling salesman problem I: general tools and comb separation. Mathematical Programming 2002;92:237–55.
[38] Wolsey L. Integer programming. John Wiley and Sons; 1998.
[39] Martinez LC, da Cunha AS. Um Arcabouço Local Branching para Problemas de Otimização Combinatória aplicado ao Problema da Árvore de Custo Mínimo com *k* arestas, XLI Simpósio Brasileiro de Pesquisa Operacional; 2009 [in Portuguese].
[40] Martinez LC, da Cunha AS. 〈http://www.dcc.ufmg.br/∼leocm/lbf〉.
[41] Danna E, Pappe CL. Branch-and-price heuristics: a case study on the vehicle routing problem with time windows. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column generation. Springer; 2005.
[42] Chen Z, Xu H. Dynamic column generation for dynamic vehicle routing with time windows. Transportation Science 2006;40(1):74–88.
[43] Hoos H, Stützle T. Stochastic local search—foundations and applications. Elsevier; 2004.
[44] Julstrom BA, Coding TSP. Tours as permutations via an insertion heuristic. In: SAC '99: Proceedings of the 1999 ACM symposium on applied computing. San Antonio, Texas, United States: ACM Press; 1999. p. 297–301.
[45] Prais M, Ribeiro CC. Reactive grasp: an application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing 2000;12:164–76.
[46] Croes GA. A method for solving traveling-salesman problems. Operations Research 1958;6(6):791–812.
[47] Michiels W, Aarts E, Korst J. Theoretical aspects of local search. Springer; 2007.
[48] de Oliveira HCB, Vasconcelos GC, Alvarenga GB, Mesquita RV, Souza MM. A robust method for the VRPTW with multi-start simulated annealing and statistical analysis. In: SCIS '07: IEEE symposium on computational intelligence in scheduling, 2007. p. 198–205.
[49] Bentley JL. Experiments on traveling salesman heuristics. In: SODA '90: Proceedings of the first annual ACM-SIAM symposium on discrete algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 1990. p. 91–9.
[50] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–100.
[51] 〈http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95〉; 2007.
[52] ILOG Cplex Solver 〈http://www.ilog.com/products/cplex/〉.