Comput Manag Sci (2010) 7:269–287
DOI 10.1007/s10287-009-0119-2

ORIGINAL PAPER

A metaheuristic for the min–max windy rural postman problem with K vehicles

Enrique Benavent · Ángel Corberán · José M. Sanchis

Received: 22 May 2009 / Accepted: 4 December 2009 / Published online: 27 December 2009

Abstract In this paper we deal with the min–max version of the windy rural postman problem with K vehicles. For this problem, in which the objective is to minimize the length of the longest tour in order to find a set of balanced tours for the vehicles, we present here a metaheuristic that produces very good feasible solutions in reasonable computing times. It is based on the combination of a multi-start procedure with an Iterated Local Search. Extensive computational results on a large set of instances with up to 50 vertices, 184 edges and 5 vehicles are presented. The results are very good, the average gaps with respect to a known lower bound are less than 0.40% for instances with 2 or 3 vehicles and up to 1.60% when 4 or 5 vehicles are considered.

Keywords Rural postman problem · Windy rural postman problem · Metaheuristics · Multivehicles

1 Introduction

Given an undirected and connected graph, the problem of finding a minimum cost closed walk traversing each edge at least once is the well known Chinese Postman Problem (CPP). The CPP is solvable in polynomial time. However, as in some real world applications, if the cost of traversing an edge (i, j) from i to j is different to the one of traversing it from j to i, we have an NP-hard problem known as the Windy

E. Benavent · Á. Corberán (✉)
DEIO, Universitat de València, Valencia, Spain
e-mail: angel.corberan@uv.es

J. M. Sanchis
DMA, Universidad Politécnica de Valencia, Valencia, Spain

Postman Problem (WPP). The WPP was proposed by Minieka (1979), and generalizes the Chinese Postman problems defined on undirected, directed and mixed graphs.

The Windy Rural Postman Problem (WRPP) is a generalization of the WPP in which not all the edges in the graph have to be traversed but only those in a given subset of required edges. Consider an undirected graph $G = (V, E)$ with two costs c_{ij}, c_{ji} associated with each edge $e = (i, j)$, which will be called a windy graph in what follows, and let $E_R \subseteq E$ be the set of required edges. Then, the WRPP is defined as the problem of finding a minimum cost closed walk (tour) traversing every edge in E_R at least once. If, in addition, a subset of vertices $V_R \subseteq V$ has to be visited by the solution, we have the Windy General Routing Problem (WGRP).

Several papers in the literature have been devoted to the WRPP and the WGRP. Benavent et al. (2005, 2007) propose an integer linear programming formulation and different heuristics and lower bounds for the WRPP. The WRPP and WGRP polyhedra have been studied in Corberán et al. (2008) and, based on those polyhedral descriptions, a branch-and-cut capable of solving large size WRPP and WGRP instances is described in Corberán et al. (2007).

In some real world applications the goal is not to design a single tour for one vehicle but a set of several tours that jointly service the required edges. In this paper we deal with such an extension of the WRPP, the min–max K-vehicles Windy Rural Postman Problem (MM K-WRPP), that is defined as follows. Given a windy graph $G = (V, E)$, a distinguished vertex, $1 \in V$, representing the depot, a subset of required edges $E_R \subseteq E$, and a fixed number K of vehicles, the MM K-WRPP consists of finding a set of K tours for the vehicles in such a way that each tour starts and ends at the depot and each required edge is serviced by exactly one vehicle. The objective is to minimize the length of the longest tour in order to find a set of balanced routes for the vehicles. Some real-life applications of routing problems with min–max objectives are school bus routing (Delgado and Pacheco 2001), the delivery of newspapers to customers (Applegate et al. 2002) and waste collection (Lacomme et al. 2004).

The MM K-WRPP has been study in Benavent et al. (2009), where an integer linear formulation is proposed and its associated polyhedron is defined and partially described. Moreover, a branch-and-cut producing promising computational results is presented. However, the proposed branch-and-cut was not able to solve even medium size instances because good feasible solutions were hard to find during the branching process. The knowledge of a good feasible solution would reduce the size of the enumeration tree of the branch-and-cut algorithm. The aim of this work is the design and implementation of an algorithm capable of providing good MM K-WRPP feasible solutions and tight upper bounds.

Other related papers on Arc Routing Problems concerning a fleet of vehicles of unlimited capacity are those by Assad et al. (1987) and Pearn (1994). These two papers deal with the K-CPP, in which we look for K routes starting and ending at the depot such that all the edges of the graph are serviced by exactly one vehicle and the total distance is minimized. In the first paper it is shown that the undirected and directed cases can be solvable in polynomial time, while in the second Pearn proved that the K-CPP is NP-hard when it is defined on a mixed or windy graph. The min–max K-CPP was introduced in Frederickson et al. (1978). Authors proved that the min–max K-CPP is NP-hard and proposed a $(2-\frac{1}{K})$-approximation algorithm.

More recently, Ahr and Reinelt present several lower bounds and heuristics for this problem (Ahr and Reinelt 2002) and a Tabu Search procedure that produces very good solutions (Ahr and Reinelt 2006). In Ahr (2004) some more results on the min–max K-CPP, including an exact solution method based on a branch-and-cut approach, are presented. Finally, let us mention that (Lacomme et al. 2004) also tackled a min–max objective on the Capacitated Arc Routing Problem.

This paper is organized as follows. In Sect. 2 the definition and the formulation of the problem proposed in Benavent et al. (2009) is presented. Also the notation used and the main known results for the MM K-WRPP are there summarized. Section 3 describes the metaheuristic we propose for the MM K-WRPP resolution. Finally, the computational results obtained on a large set of instances are shown in Sect. 4.

2 Min–max K-vehicles windy RPP

As said in the Introduction, the MM K-WRPP consists of finding a tour (closed walk starting and ending at the depot) for each vehicle such that each required edge is traversed at least once by at least one vehicle. Such a set of tours is called a K-WRPP solution. The objective is to find a K-WRPP solution such that the cost of the maximum cost vehicle tour is minimum. The maximum cost vehicle tour will also be called the longest tour throughout the paper.

In what follows, and for the sake of simplicity, we assume that each vertex in V is incident with at least one required edge. This is not a restriction as there exists a simple way to transform an instance not satisfying this assumption into an equivalent one which does (see, e.g.,Christofides et al. (1986) or Eiselt et al. (1995)). Given $S, S' \subseteq V$, $(S : S')$ denotes the edge set with one end-point in S and the other in S', while for a node subset, $S \subseteq V$, $\delta(S) = (S : V \backslash S)$ and $E(S) = \{(i, j) \in E : i, j \in S\}$. $\delta_R(S)$, $E_R(S)$ and $(S : S')_R$ denote the previous sets restricted to the required edges.

To formulate the problem, Benavent et al. (2009) define $2K$ variables x_{ij}^k and x_{ji}^k, associated with each edge $e = (i, j) \in E$, representing the number of times edge e is traversed by vehicle k from i to j or from j to i, respectively. If edge e is required, K more variables y_{ij}^k which take the value 1 if edge e is serviced by the vehicle k and 0 otherwise, are also defined. Finally, an artificial variable z is used to minimize the maximum tour cost.

For any subset $F \subseteq E$, $x^k(F)$ denotes $\sum_{(i,j)\in F} (x_{ij}^k + x_{ji}^k)$. The proposed MM K-WRPP formulation is:

$$\text{Minimize} \quad z$$

s.t.:

$$\sum_{(i,j)\in E} \left(c_{ij} x_{ij}^k + c_{ji} x_{ji}^k\right) \leq z \quad k = 1, \ldots, K \tag{1}$$

$$\sum_{k=1}^{K} y_e^k = 1, \quad \forall e \in E_R \tag{2}$$

$$x_{ij}^k + x_{ji}^k \geq y_e^k \quad \forall e = (i,j) \in E_R\ ,\ k = 1, \ldots, K \tag{3}$$

$$\sum_{(i,j)\in\delta(i)} (x_{ij}^k - x_{ji}^k) = 0, \forall i \in V\ ,\ k = 1, \ldots, K \tag{4}$$

$$x^k(\delta(S)) \geq 2y_e^k, \forall S \subset V \setminus \{1\} \text{ with } |E_R(S)| \geq 1,$$
$$\forall e \in E_R(S), \quad k = 1, \ldots, K \tag{5}$$

$$x_{ij}^k, x_{ji}^k \geq 0 \text{ and integer } \forall (i,j) \in E\ ,\ k = 1, \ldots, K \tag{6}$$

$$y_e^k \in \{0,1\} \forall e \in E_R\ ,\ k = 1, \ldots, K \tag{7}$$

Inequalities (1) imply that the maximum cost vehicle route is minimized. Equations (2) assure that each required edge is serviced by exactly one vehicle and inequalities (3), called *traversing inequalities*, force a vehicle to traverse the edges it services. *Symmetry equations* (4) force each vehicle tour to be symmetric, while *connectivity inequalities* (5) ensure that each tour connects the edges it services and the depot.

A solution for the MM K-WRPP on G is a vector $(x^1, y^1, x^2, y^2, \ldots, x^K, y^K)$ with $(2|E| + |E_R|)K$ components satisfying (2) to (7). Given a vehicle k, the pair (x^k, y^k) and the vector x^k are called *route* and *tour*, respectively. As noted in Benavent et al. (2009), this formulation allows solutions in which a vehicle tour x^k is formed by several disconnected subtours, one of them connecting all the edges it services to the depot and the others traversing edges not serviced by this vehicle. Note also that solutions where a given vehicle neither traverses nor services any edges are also allowed ($x^k = y^k = 0$).

Given a K-WRPP solution $(x^1, y^1, x^2, y^2, \ldots, x^K, y^K)$, it is easy to see that the sum of all the vehicle tours x^k produces a WRPP tour $x = \sum x^k$. Then, as it is pointed out in Benavent et al. (2009), from every known family of valid inequalities for the WRPP (see Corberán et al. 2008), valid inequalities for the MM K-WRPP are obtained. These are called aggregate inequalities. For instance, the aggregate R-odd cut inequalities are:

$$\sum_{k=1}^{K} x^k(\delta(S)) \geq |\delta_R(S)| + 1, \quad \forall S \subset V \quad \text{such that } |\delta_R(S)| \text{ is odd} \tag{8}$$

and are based on the fact that any MM K-WRPP solution must cross any given edge cutset an even number of times. On the other hand, when a single vehicle k is considered, all the edges $e \in E_R$ are not necessarily 'required' for vehicle k because it could either not service nor traverse e. The actual required edges for a given vehicle k are those determined by the vector y^k. This is the idea behind the disaggregate inequalities. For example, the disaggregate version of the R-odd inequalities involving variables y^k is as follows. Let $\delta(S)$ be an edge cutset on G and let $F \subset \delta_R(S)$ be a subset of required edges with $|F|$ odd. For each vehicle k, the following inequality

$$x^k(\delta(S)) \geq 2y^k(F) - |F| + 1, \tag{9}$$

is called parity or cocircuit inequality (Barahona and Grötschel 1986) and is valid for the MM K-WRPP.

Above inequalities can be generalized from a single vehicle to several ones. If all the edges in $F \subset \delta_R(S)$ are serviced by a subset of vehicles then these vehicles have to jointly traverse the cutset at least $|F|+1$ times. Again, let $F \subset \delta_R(S)$ be a subset of required edges with $|F|$ odd. For each subset $\{k_1, k_2, \ldots, k_P\}$ with $P < K$ vehicles, the inequality

$$x^{k_1}(\delta(S)) + \cdots + x^{k_P}(\delta(S)) \geq 2y^{k_1}(F) + \cdots + 2y^{k_P}(F) - |F| + 1, \qquad (10)$$

is valid for the MM K-WRPP and is called P-aggregate parity inequality.

The above ideas about obtaining disaggregate and P-aggregate inequalities from valid inequalities for the (1 vehicle) WRPP have been also applied to K-C, Honeycomb, Path-Bridge and other inequalities (Benavent et al. 2010). Moreover, in Benavent et al. (2009) it is shown that, under mild conditions, trivial, traversing (3), connectivity (5) and all the above commented inequalities for the MM K-WRPP also define facets of its associated polyhedron.

On the other hand, let us suppose that we know an upper bound ub for the MM M-WRPP. Let $e_1, e_2, \ldots, e_p$ be a set of required edges and let $z(WGRP)$ be the optimal cost of the Windy General Routing Problem defined on the graph induced by these edges plus the depot. If $z(WGRP) > ub$ holds, then, in any optimal MM M-WRPP solution, a single vehicle cannot service all the edges $e_1, e_2, \ldots, e_p$, and the inequalities

$$y^k_{e_1} + y^k_{e_2} + \cdots + y^k_{e_p} \leq p - 1, \quad \forall \text{ vehicle } k \qquad (11)$$

have to be satisfied by any optimal solution. Moreover, let S be the set of vertices incident with edges $e_1, e_2, \ldots, e_p$. Then, since at least two vehicles have to enter into S, any optimal solution satisfies the following inequality:

$$\sum_{k=1}^{M} x^k(\delta(S)) \geq 4. \qquad (12)$$

Inequalities (12) can be generalized to $\sum_{k=1}^{M} x^k(\delta(S)) \geq 2\lceil z(WGRP)/ub \rceil$. Their usefulness strongly depends on the quality of the upper bound ub.

3 A metaheuristic for the MM K-WRPP

In this section we describe a metaheuristic for the MM K-WRPP resolution whose main components are a Multi-Start (MS) algorithm, a Variable Neighborhood Descent (VND) and an Iterated Local Search (ILS) procedure. It is based on the Multi-Start Iterated Local Search algorithm proposed by Belenguer et al. (2009) for the Split Delivery Capacitated Arc Routing Problem and can be summarized as follows.

Overall the algorithm works as follows, the multi-start algorithm described in Benavent et al. (2005) produces, at each iteration, a single route traversing all the required edges (a feasible solution for the WRPP on graph G), which is then split into K small routes, one for each vehicle, to obtain a MM K-WRPP solution. Then, two different VNDs are applied to this solution. The first VND uses moves that try to improve each route independently (WRPP moves), while the second one uses inter route moves.

The improved solution is used as the starting solution of the ILS procedure. At any iteration of the ILS, the MM K-WRPP solution is perturbed in two steps: the K routes are merged to form a giant tour, which is randomly perturbed and then split again into K routes. The resulting MM K-WRPP solution is then improved using similar procedures to those described above. The best solution found during the ILS is used as the starting solution for the next iteration. The best solution found in the whole MS procedure is the output of the algorithm. All these procedures are described in detail in the following sections.

3.1 Initial WRPP solutions

In the first step of the algorithm, a number *nsol* of WRPP solutions are generated. The first five initial solutions are obtained with the constructive heuristics WRPP1, WRPP2 and WRPP3 presented in Benavent et al. (2007) and with its variants WRPP4 and WRPP5 proposed in Benavent et al. (2005). The remaining *nsol* -5 WRPP solutions are generated with the multi-start algorithm presented in Benavent et al. (2005). All these WRPP heuristics are briefly described below.

ALGORITHM WRPP1

This algorithm consists of three phases. First, some edges are added to the graph induced by the required edges, G_R, to obtain a new connected graph G'_R. The edges added are those corresponding to a minimum cost tree spanning the connected components of G_R. Second, a minimum cost matching problem on the odd degree nodes of graph G'_R is solved. Edges in the matching solution are also added to G'_R to obtain an even and connected graph G''_R. Third, Win's exact algorithm (Win 1989) for the WPP defined on Eulerian graphs is then applied on G''_R to obtain a feasible solution to the WRPP on G.

ALGORITHM WRPP2

Basically, this algorithm executes the same phases of *WRPP1* but in a different order. It begins by assigning to each edge in G_R the direction associated with its minimal traversing cost to obtain a directed graph, G^d_R. From G^d_R a new balanced mixed graph, G^1, is obtained by solving a minimum cost flow problem with demands $d(i)$, for each node i, computed as the difference between the arcs in G^d_R entering at and leaving from i. Then, a minimum cost matching on the odd degree nodes of G^1 is solved. Edges in the matching solution are added to G^1 to obtain an even mixed graph G^2. Again, edges in G^2 are oriented in the direction of their minimal traversing cost, and a new minimum cost flow problem is solved to obtain a symmetric digraph G^3. Finally, as graph G^3 may be disconnected, a last phase consisting of the resolution of a shortest spanning tree connecting its components is executed.

ALGORITHM WRPP3

This is a simple heuristic that, first, computes a shortest spanning tree connecting all the components of graph G_R. The edges in the SST solution are added to G_R to obtain an undirected and connected graph G'. Second, the edges in G' are oriented according

to the minimum traversal cost, and demands $d(i)$ for each node i are computed as in heuristic *WRPP2*. A Transportation Problem is then solved and copies of the corresponding arcs are added to G' to obtain a connected and symmetric directed graph, which is a feasible WRPP solution.

ALGORITHM WRPP4

This constructive heuristic is a modification of algorithm *WRPP1*. As in the *Modified Mixed Algorithm* for the MCPP (Raghavachari and Veerasamy 1998), the idea of algorithm *WRPP4* is to try to anticipate during the first steps what will happen in the last phases of the algorithm.

First, the algorithm connects the components of G_R by adding the edges of a shortest spanning tree computed as in the first phase of algorithm *WRPP1*. Again, $G'_R = (V, E'_R)$ represents the resulting graph. Now, before solving a minimum cost matching problem to make G'_R even, the edge costs are changed in such a way the solution to the matching problem includes, at least partially, the solution to the flow problem of the last phase.

In order to do this, first, the *average cost* C_a of the edges in E'_R is computed as $C_a = \frac{1}{|E'_R|} \sum_{ij \in E'_R} \frac{c_{ij}+c_{ji}}{2}$. Now, the sets $E_1 = \{(i, j) \in E'_R : |c_{ji} - c_{ij}| > s * C_a\}$ and $E_2 = E \setminus E_1$ are defined, where s is a positive parameter that has been fixed to 0.2. Then, a directed graph G^d_R is constructed with set of nodes V and an arc (i, j) for each edge $(i, j) \in E_1$ (it is assumed that $c_{ij} \leq c_{ji}$). Note that the set of arcs in G^d_R is associated with the edges in G'_R whose traversal is quite more expensive in one direction than in its opposite one. The guess is that, in the final solution, these edges will be traversed in the direction of their minimal cost.

A minimum cost flow problem, with demands $d(i)$ computed in graph G^d_R, is now solved on an auxiliary digraph $G_{aux} = (V, A_{aux})$. A_{aux} contains two arcs (i, j) and (j, i), with costs c_{ij} and c_{ji}, respectively, and infinite capacity, for each edge of the original graph G. Furthermore, for the edges $(i, j) \in E_1$, another arc (j, i) with cost $\frac{c_{ji}-c_{ij}}{2}$ and capacity 2 is added to A_{aux}. Those edges $(i, j) \in E_1$ that are traversed by at least one unit flow in one of its two possible directions are added to a list L. This list also contains the edges $(i, j) \in E_2$ such that the flow through them is at least of two units in one of its two possible directions. Edges belonging to this list are the ones that seem to be good candidates to appear in the solution and whose costs will be modified in the next phase of the algorithm.

The heuristic now proceeds to make graph G'_R even. To do this, a minimum cost matching problem is solved in which edges in L have zero cost and the other edges have original costs. Edges in the resulting graph are finally oriented by solving a minimum cost flow problem with the original costs for all the edges.

ALGORITHM WRPP5

This other constructive heuristic is a modification of algorithm *WRPP2*. Again, the idea is to guess which arcs will be added to the solution in the last phase of the procedure in order to introduce them in the solution during the previous steps.

Algorithm *WRPP5* starts computing a SST connecting the components of graph G_R. However, the edges in the shortest path in G associated with each edge in the

spanning tree are not added here to G_R, but listed in L and their average cost is computed. For each edge $(i, j) \in E$ (it is assumed that $c_{ij} \leq c_{ji}$), new costs c'_{ij} and c'_{ji} are now defined as follows:

- $c'_{ij} = c_{ij} - \alpha C_a$ and $c'_{ji} = c_{ji} - \alpha C_a$, if i and j are in different connected components of G_R and $c_{ij} \geq \alpha C_a$.
- $c'_{ij} = 0$ and $c'_{ji} = c_{ji} - c_{ij}$, if i and j are in different connected components of G_R and $c_{ij} < \alpha C_a$.
- $c'_{ij} = c_{ij}$ and $c'_{ji} = c_{ji}$, otherwise,

where α is a parameter that has been fixed to 0.8.

Algorithm *WRPP2* is then executed with these new costs. Only the computation of the shortest spanning tree during the last phase of the algorithm, if needed (when G^3 is not a connected graph), is performed with the original costs c_{ij}.

WRPP Multi- Start algorithm

In Benavent et al. (2005) several multi-start (MS) heuristics designed from the constructive algorithms described above are presented. Basically, multi-start algorithms consist of the execution of a number of global iterations until some stopping criterium is satisfied. Each global iteration generates a solution with a constructive algorithm and then improves it with a local search method.

Since each one of the constructive heuristics described above produces only one solution, Benavent et al. introduce several random elements to obtain a set of different feasible solutions with each algorithm. To get solutions as different from each other as possible, the random elements are introduced in the first phases of the algorithms. As the best results were reported (Benavent et al. 2005) with the MS heuristic based on algorithm WRPP2, this is the one we have used here as WRPP multi-start algorithm.

The randomization strategy consists of modifying the cost of the shortest paths. This cost modification should be large enough to get different solutions, but not as large as to produce bad solutions. In this case, the cost $\hat{c}_{ij}$ of the shortest path between every pair of nodes i, j is substituted by a new cost randomly chosen in the interval $[0.6\hat{c}_{ij}, 1.4\hat{c}_{ij}]$, i.e. with a maximum deviation of 40% from the original cost. With this strategy, other *nsol - 5* WRPP solutions are generated.

Variable Neighborhood Descent for the WRPP

Several improvement methods were applied in Benavent et al. (2005) to each WRPP solution. We have used here these improvement methods to build a variable neighborhood descent procedure that is applied to WRPP solutions and to each route of the MM K-WRPP solutions.

Variable Neighborhood Descent (VND) Mladenovic and Hansen (1997) is an enhanced local improvement strategy based on a sequence $(N_1, \ldots, N_T)$ of T neighborhoods with growing cardinals. Starting from $k = 1$, each VND iteration searches the neighborhood N_k. If one improving move is detected, it is executed and k is reset to 1, otherwise k is incremented. The method stops when $k = T$ and the exploration of N_T brings no improvement. In our algorithm, $T = 3$ neighborhoods, associated with the 3 procedures described below, are used.

A WRPP solution is encoded as a sequence of the required edges, each one traversed in a given direction, and it is assumed that the route follows the shortest path from the final vertex of any required edge in the sequence to the initial vertex of the next one. The first improvement procedure, called *2-interchange*, is a steepest-descent algorithm in which a simple move consisting of interchanging the positions of two required edges in the sequence they are serviced is performed at each iteration. The second procedure is also a steepest-descent algorithm in which each move is an *Or-interchange* (Or 1976). In this move, a section consisting of at most L consecutive required edges in the sequence is inserted elsewhere between two consecutive required edges, the first one of which must be among the M edges closest to the first edge of the section. This insertion is performed in such a way that the direction of traversal remains unchanged. As in Benavent et al. (2005), the chosen values for L and M are 4 and 11, respectively. The third improvement procedure finds the optimal direction in which each required edge has to be traversed for a given order of traversal of the required edges. It is called the *reversal procedure* and has been proposed in Lacomme et al. (2002) and in Ramdane-Chérif (2002). See Benavent et al. (2005) for details.

In our VND the first searched neighborhood is defined by the Or-interchange move. The second one corresponds to the reversal procedure, and the last one is defined by the 2-interchange move.

3.2 Split procedure

Remember that a WRPP solution is encoded as a sequence of the required edges, each one traversed in a given direction. This sequence can be split into K routes applying the procedure described in Lacomme et al. (2004) to obtain a MM K-WRPP solution.

Given a sequence of the required edges $(S_1, S_2, \ldots, S_m)$, where $m = |E_R|$, the algorithm starts by building an auxiliary directed graph H with $m+1$ vertices indexed from 0 to m. An arc $(i-1, j)$ in H corresponds to a route starting at the depot, traversing the required edges in the subsequence $(S_i, \ldots, S_j)$ in this order and coming back to the depot. The arc weight is given by the route cost. Now, the path between vertices 0 and m, which minimizes the cost of the largest arc weight and contains at most K arcs, is computed. The algorithm proposed by Lacomme et al. (2004) finds this path in $O(min(m, K) \cdot m^2)$.

3.3 Improvement procedures for the MM K-WRPP

We describe here two improvement procedures that are used on MM K-WRPP solutions. The first one is a simplification method that is executed on an expanded representation of the whole MM K-WRPP solution; the second one is a VND procedure based on several moves among the routes.

Simplification procedure

The simplification procedure starts by substituting each shortest path between any two consecutive required edges in the routes of the MM K-WRPP solution by the

corresponding sequence of edges. Then, each route is built as a closed walk, i.e., a sequence of required and non required edges traversed in a given direction that starts and ends at the depot. We call *expanded representation* this way of representing a route in contrast with the *compact representation* consisting of a sequence of required edges only. Note that such a closed walk can be represented by a directed graph that contains as many copies of a given arc as the number of times it is traversed by the closed walk. It is well known that this graph is connected and symmetric, and, viceversa, given a connected and symmetric graph there is a closed walk traversing each arc exactly once (also called Eulerian walk).

Let us assume, for instance, that a route traverses a given edge (i, j) twice from i to j and three times from j to i. Then, we can remove two arcs (i, j) and two arcs (j, i) from the corresponding graph, and the resulting graph is still connected and symmetric, so there is a route with cost no greater than the original one covering the same set of edges. The simplification procedure uses this idea to remove any extra traversal of the edges and works as follows.

For each route and each edge $e = (i, j)$, we compute the number of times the route traverses the edge e in each direction, say x_{ij} and x_{ji}, and let

$$\delta = \begin{cases} \min\{x_{ij}, x_{ji}\} & \text{if } x_{ij} \neq x_{ji}, \\ x_{ij} - 1 & \text{if } x_{ij} = x_{ji}. \end{cases}$$

If $\delta > 0$, we remove δ arcs from i to j and from j to i. After these simplifications the resulting graph is connected and symmetric, and an Eulerian walk is built thus obtaining an improved route.

Once all the routes are simplified in this way, the longest route is determined and the service of required edges is assigned to the other routes, as long they are traversed by them. Then, the service of the remaining required edges is assigned to the longest route and it is then improved by computing the shortest paths between any two consecutive required edges whose service has been assigned to it. These operations may result in a decrease of its cost, thus making it possible that this route is no longer the longest one. In this case, the procedure is applied again to the new longest route and repeated until no improvement is achieved.

VND USING MOVES AMONG THE ROUTES

We now describe three moves that involve two routes in any MM K-WRPP solution represented in compact form. The first two moves consist of moving one or two consecutive required edges from the longest route to another route, while in the third move one required edge of the longest route and another one of another route are interchanged. These moves are applied in the procedures *change1to0*, *change2to0* and *change1to1*, respectively. All of them use the following strategy: all the possible moves are searched but once an improving move is found, it is executed and the procedure stops.

A Variable Neighborhood Descent has been implemented using the above procedures in the following order: *change2to0*, *change1to0* and *change1to1*.

3.4 The iterated local search algorithm

ILS (see Lourenço et al. 2002) is a metaheuristic that uses an initial solution, a local search and a perturbation procedure. Its structure is as follows. Given an initial solution, it is improved by local search. Then, the main loop of the algorithm performs *niterILS* iterations. At each iteration, a copy of the current best solution, called *BestSol*, is randomly modified using a *perturbation procedure*. The resulting solution is also improved by local search and the best solution is updated.

The perturbation procedure works as follows. From the K routes in the MM K-WRPP solution, a global tour for the WRPP is built just joining the sequences of required edges in the compact representation of the routes. Then, the WRPP tour is perturbed by interchanging two randomly selected required edges. The new WRPP tour is split into K routes to get a different MM K-WRPP solution by using the splitting procedure described above.

The local search we use in the ILS has the following components. First, the VND procedure for the WRPP (called VND1 for short) is applied to every route of the solution. Then, the *simplification* procedure described in Sect. 3.3 is executed and is followed by the application of the VND that uses moves among the routes (denoted by VND2).

3.5 The metaheuristic

The metaheuristic we propose for the MM K-WRPP uses all the procedures described above combined in the following way.

Algorithm 1 – Pseudo-code of the multi-start ILS metaheuristic

```
for iterMS := 1 to nsol do
  generate(WRPPsol)
  VND1(WRPPsol)
  split(WRPPsol,K-WRPPsol)
  for i := 1 to K do
    VND1(K-WRPPsol(i))
  end for
  Simplification(K-WRPPsol)
  VND2(K-WRPPsol)
  BestSol := K-WRPPsol
  UpdateBestSolution(BestSol,GlobalBestSol)
  for iterILS := 1 to niterILS do
    SolILS := perturb(BestSol)
    for i := 1 to K do
      VND1(SolILS(i))
    end for
    Simplification(SolILS)
    VND2(SolILS)
    UpdateBestSolution(SolILS,BestSol)
  end for
  UpdateBestSolution(BestSol,GlobalBestSol)
end for
Return(GlobalBestSol)
```

Table 1 Characteristics of the 144 MM K-WRPP instances

	# of instances	$\|V\|$	$\|E\|$	$\|E_R\|$	# of R-sets
C01	6	11	13	7	4
C02	6	14	33	12	4
C03	6	28	57	26	4
C04	6	17	35	22	3
C05	6	20	35	16	5
C06	6	24	46	20	7
C07	6	23	47	24	3
C08	6	17	40	24	2
C09	6	14	26	14	3
C10	6	12	20	10	4
C11	6	9	14	7	3
C12	6	7	18	5	3
C13	6	7	10	4	3
C14	6	28	79	31	6
C15	6	26	37	19	8
C16	6	31	94	34	7
C17	6	19	44	17	5
C18	6	23	37	16	8
C19	6	33	54	29	7
C20	6	50	98	63	7
C21	6	49	110	67	6
C22	6	50	184	74	6
C23	6	50	158	78	6
C24	6	41	125	55	7
Average		25.1	59.0	28.1	5.0

At each iteration of the multi-start algorithm, a WRPP solution is generated using the algorithms described in Sect. 3.1. VND1 is applied to each WRPP solution and the resulting tour is split to get a MM K-WRPP solution. VND1 is applied to every route and then the improvement procedures *simplification* and VND2 are executed. The resulting MM K-WRPP solution, if it is different to the ones already generated, is used as the starting solution of the ILS procedure.

Algorithm 1 summarizes our multi-start ILS metaheuristic, in which UpdateBestSolution(A,B) means that solution B is updated with solution A if the latter is better in terms of the objective function.

4 Computational results

We present here the computational results obtained with our metaheuristic on a large set of instances in the Literature. The algorithm has been coded in C/C++ and all the tests were run on an Intel Core 2 CPU 2.40 GHz and 2GB RAM.

Table 2 Computational results on the MM 2-WRPP instances

	Opt.	(H1)			(H2)			(H3)			(H4)		
		Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.
C01	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C02	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C03	6	0.0	6	5.8	0.0	6	1.9	0.0	6	1.9	0.1	5	2.2
C04	6	0.1	6	4.0	0.1	6	1.2	0.2	4	0.8	0.2	4	1.0
C05	6	0.0	6	0.3	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C06	6	0.0	6	2.3	0.0	6	0.8	0.2	5	0.6	0.2	5	0.8
C07	6	0.0	6	5.1	0.0	6	1.7	0.0	6	1.2	0.0	6	1.4
C08	6	0.0	6	6.6	0.2	5	2.2	0.2	5	1.5	0.4	5	1.7
C09	6	0.0	6	0.4	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C10	6	0.0	6	0.1	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C11	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C12	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C13	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C14	6	0.0	6	18.3	0.0	6	5.4	0.1	4	4.0	0.2	4	4.5
C15	6	0.0	6	1.6	0.0	6	0.6	0.0	6	0.4	0.0	6	0.5
C16	6	0.1	5	17.8	0.0	5	5.5	0.1	4	4.5	0.1	4	5.4
C17	6	0.0	5	1.0	0.0	6	0.3	0.0	6	0.3	0.0	6	0.4
C18	6	0.0	6	1.2	0.0	6	0.4	0.0	6	0.3	0.0	6	0.4
C19	6	0.0	6	11.2	0.0	6	3.4	0.0	6	2.7	0.0	6	3.0
C20	6	0.3	5	175.0	0.4	4	49.1	0.5	3	61.5	0.5	3	90.2
C21	6	0.9	5	214.8	2.1	0	60.7	1.1	3	77.0	1.0	4	122.2
C22	6	0.3	3	351.6	0.7	0	95.9	0.7	0	135.0	0.5	3	231.4
C23	6	0.5	5	404.0	1.2	1	110.5	0.9	3	153.8	0.7	3	293.9
C24	6	0.2	6	128.0	0.6	1	35.6	0.3	3	40.7	0.3	3	52.2
Av.1		0.1	5.7	56.2	0.2	4.9	15.6	0.2	4.9	20.3	0.2	5.0	33.8
Av.2		0.4	4.8	254.7	1.0	1.2	70.4	0.7	2.4	93.6	0.6	3.2	158.0

The tested MM K-WRPP instances are those used in Benavent et al. (2009), that were obtained from the WRPP instances presented in Benavent et al. (2007), considering vertex 1 as the depot and different numbers of vehicles, from 2 to 5. The 144 WRPP instances were built by randomly generating 6 instances from each of the 24 undirected RPP instances proposed in Christofides et al. (1981). They have up to 50 vertices, 184 edges, 78 required edges and 8 R-sets (vertex sets of the connected components induced by the required edges). The whole set of MM K-WRPP instances can be found in http://www.uv.es/corberan, as well as the optimal value (if known) or a lower bound obtained with the algorithm presented in Benavent et al. (2010) for each instance and each number of vehicles.

Table 1 shows the characteristics of the instances. First column shows the name of the set of 6 WRPP instances corresponding to the original RPP instance generated by

Table 3 Computational results on the MM 3-WRPP instances

	Opt.	(H1)			(H2)			(H3)			(H4)		
		Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.
C01	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C02	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.1
C03	6	0.0	6	4.0	0.0	6	1.3	0.0	6	1.2	0.0	6	1.5
C04	6	0.0	6	2.7	0.2	4	0.8	0.2	4	0.8	0.2	3	0.9
C05	6	0.0	6	0.3	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C06	6	0.0	6	1.9	0.0	6	0.6	0.4	5	0.6	0.4	5	0.8
C07	6	0.0	6	3.1	0.2	5	1.0	0.1	5	1.1	0.1	5	1.2
C08	6	0.0	6	5.0	0.1	5	1.6	0.0	6	1.4	0.0	6	1.6
C09	6	0.0	6	0.3	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C10	6	0.0	6	0.1	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C11	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C12	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C13	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C14	6	0.0	6	10.5	0.1	5	3.1	0.5	4	3.1	0.5	4	3.3
C15	6	0.0	6	1.2	0.0	6	0.4	0.0	6	0.4	0.0	6	0.4
C16	6	0.1	6	10.2	0.3	4	3.4	0.3	4	3.7	0.3	4	4.0
C17	6	0.0	6	0.8	0.0	6	0.2	0.0	6	0.3	0.0	6	0.4
C18	6	0.0	6	1.1	0.0	6	0.3	0.0	6	0.3	0.0	6	0.4
C19	6	0.0	6	6.3	0.2	5	1.9	0.2	5	2.1	0.2	5	2.3
C20	4	0.8	5	79.4	0.9	3	24.7	0.9	4	37.1	0.6	6	50.1
C21	2	1.6	4	97.5	2.0	2	29.6	2.0	2	44.1	1.5	4	67.2
C22	0	1.5	4	147.1	1.8	2	44.7	1.8	1	72.2	1.7	2	117.9
C23	1	2.1	5	162.5	2.6	2	49.7	2.3	3	76.0	2.5	2	142.3
C24	3	0.7	5	59.9	1.0	3	18.1	1.2	2	24.8	1.0	2	30.1
Av.1		0.3	5.7	24.7	0.4	4.9	7.6	0.4	4.9	11.2	0.4	5.0	17.7
Av.2		1.3	4.6	109.3	1.7	2.4	33.3	1.6	2.4	50.9	1.5	3.2	81.5

Christofides et al. (1981). Columns 3, 4, 5 and 6 present the number of vertices, edges, required edges and R-sets for all the instances in each set, respectively.

The metaheuristic has been tested with four different settings for parameters *nsol* and *niterILS*. The first setting, denoted by H1, corresponds to the values (200, 200) for parameters *nsol* and *niterILS*, respectively. Settings H2 and H3 are associated with values (100, 100) and (100, 200), respectively. Finally, in H4, the values for parameters *nsol* and *niterILS* are set to $2 \cdot |E_R|$, but never less than 20 nor more than 200.

Tables 2, 3, 4, 5 show the results obtained with the different settings of the algorithm parameters on the instances with 2, 3, 4 and 5 vehicles, respectively. Only one run was performed for each instance. Each table presents, for each setting H1 to H4, three columns showing the average percentage gap among the cost of the solution provided by the algorithm and the optimum value (if known) or a lower bound, the number

Table 4 Computational results on the MM 4-WRPP instances

	Opt.	(H1)			(H2)			(H3)			(H4)		
		Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.
C01	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C02	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.1
C03	6	0.0	6	3.5	0.0	6	1.1	0.1	5	1.3	0.1	5	1.5
C04	6	0.0	6	2.6	0.0	6	0.8	0.0	6	0.8	0.0	6	0.9
C05	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C06	6	0.0	6	1.7	0.0	6	0.5	0.0	6	0.6	0.0	6	0.8
C07	6	0.0	6	2.7	0.0	5	0.9	0.0	5	1.1	0.0	5	1.2
C08	5	0.0	6	4.7	0.0	6	1.5	0.1	5	1.5	0.1	5	1.6
C09	6	0.0	6	0.3	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C10	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C11	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C12	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C13	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C14	5	0.1	6	9.0	0.2	5	2.7	0.3	5	2.9	0.3	5	3.1
C15	6	0.0	6	1.2	0.0	6	0.4	0.0	6	0.4	0.0	6	0.4
C16	3	2.1	6	8.8	2.4	3	2.9	2.8	2	3.6	2.4	4	3.8
C17	6	0.0	6	0.8	0.0	6	0.2	0.0	6	0.3	0.0	6	0.4
C18	6	0.0	6	1.0	0.0	6	0.3	0.0	6	0.3	0.0	6	0.4
C19	6	0.0	6	5.2	0.3	5	1.6	0.3	4	1.9	0.3	4	2.1
C20	1	1.3	6	56.7	2.3	2	19.0	1.8	2	31.2	1.8	2	41.3
C21	0	2.4	6	68.7	3.8	0	22.9	2.9	3	37.4	2.9	3	54.6
C22	0	3.9	6	95.4	4.8	0	32.1	4.5	1	56.7	4.3	2	88.6
C23	0	4.9	4	105.9	5.5	1	36.1	5.1	2	61.4	4.8	4	108.8
C24	0	3.7	5	43.8	4.6	1	14.0	4.4	1	21.1	4.2	2	24.8
Av.1		0.8	5.9	17.2	1.0	4.7	5.7	0.9	4.7	9.3	0.9	5.0	13.9
Av.2		3.3	5.4	74.1	4.2	0.8	24.8	3.8	1.8	41.6	3.6	2.6	63.6

of times the heuristic gives the best solution among the four settings and the average computing time in seconds. Each row corresponds to a group of 6 instances associated with the same RPP original instance and share the same underlying graph. Moreover, the last but one row shows the average results on all the instances, while the last one presents the average results for the largest ones, i.e. those instances with more than 50 required edges, which corresponds to groups C20 to C24. Finally, column *opt.* gives the number of optimal solutions known for each group of instances.

In our opinion, the obtained results are very good. Table 2 shows that, in all the instances with 2 vehicles and for all the settings, the gaps are very small. However, note that in what refers to the most difficult instances (those in groups C20–C24) there are significative differences among the behavior of the different settings. For instance, the average gap reached by H2 is 1%, more than twice the one obtained with

Table 5 Computational results on the MM 5-WRPP instances

	Opt.	(H1)			(H2)			(H3)			(H4)		
		Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.	Gap	Best	Sec.
C01	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C02	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.1
C03	6	0.0	6	3.4	0.0	6	1.1	0.0	6	1.2	0.0	6	1.4
C04	5	0.3	6	2.6	0.3	6	0.8	0.6	5	0.8	0.6	5	0.9
C05	6	0.0	6	0.2	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C06	5	1.2	6	1.7	1.2	6	0.5	1.2	6	0.6	1.2	6	0.8
C07	5	0.1	6	2.9	0.1	6	1.0	0.2	5	1.1	0.2	5	1.2
C08	4	1.5	6	4.5	1.5	6	1.5	1.5	6	1.5	1.5	6	1.6
C09	6	0.0	6	0.3	0.0	6	0.1	0.0	6	0.1	0.0	6	0.2
C10	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C11	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C12	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C13	6	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0	0.0	6	0.0
C14	4	0.9	5	8.4	0.9	5	2.6	1.1	3	2.9	1.1	3	3.0
C15	6	0.0	6	1.2	0.0	6	0.4	0.0	6	0.4	0.0	6	0.5
C16	1	4.0	5	8.4	3.9	5	2.8	4.3	4	3.6	4.3	4	3.9
C17	6	0.0	6	0.8	0.0	6	0.2	0.0	6	0.3	0.0	6	0.3
C18	6	0.0	6	1.0	0.0	6	0.3	0.0	6	0.3	0.0	6	0.4
C19	4	0.4	6	4.8	0.4	6	1.6	0.4	6	1.9	0.4	6	2.1
C20	0	3.2	6	48.9	3.8	4	17.3	3.7	2	29.0	3.4	4	38.2
C21	0	5.4	6	58.2	6.1	3	20.6	6.0	3	34.5	6.0	3	50.2
C22	0	7.3	2	77.7	7.7	1	28.1	7.3	3	51.0	7.1	4	78.8
C23	0	7.3	5	85.8	8.0	2	31.7	7.6	3	57.6	7.6	4	97.4
C24	0	3.8	5	38.2	4.5	2	12.7	4.4	2	20.0	4.2	3	23.2
Av.1		1.5	5.7	14.6	1.6	5.2	5.1	1.6	5.0	8.6	1.6	5.2	12.7
Av.2		5.4	4.8	61.7	6.0	2.4	22.1	5.80	2.6	38.4	5.6	3.6	57.6

H1, although H2 is considerably faster. Similar comments can be done for the results shown in Tables 3 to 5 for instances with 3, 4 and 5 vehicles. In all the cases, the best results have been obtained with H1, followed by H4, while H3 and H2 produce slightly worse results although they are less time consuming. However, the differences among the respective average gaps are now not so evident. Maybe this is because the gaps in the instances with 3, 4 and 5 vehicles are larger. This could be due to the fact that, since the optimal value is not known for all these instances, a lower bound has been used to compute the gap.

The average results for all the instances are also depicted in Figs. 1, 2, 3. Figure 1 shows the average gaps for the four settings of the algorithm on all the instances grouped according to the number of vehicles. We can see that the gaps for the instances with 2 or 3 vehicles are very small, less than 0.40% on average. Instances

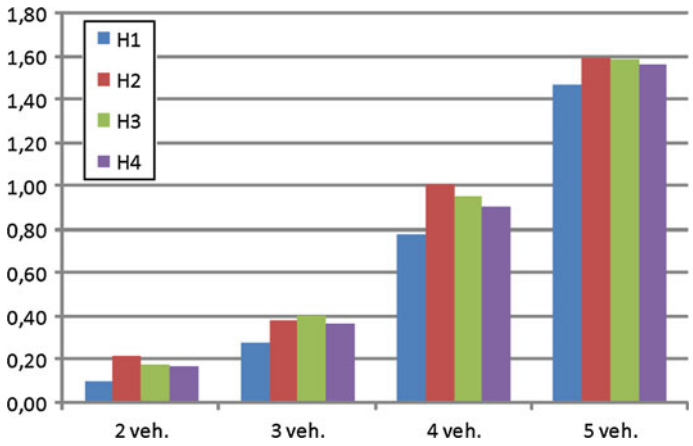

Fig. 1 Gap values

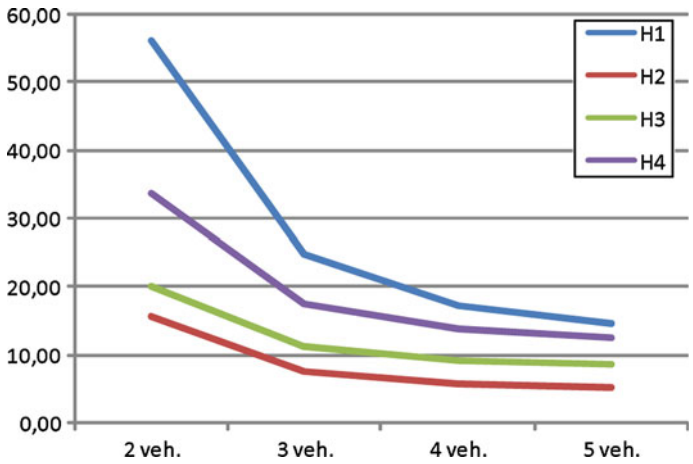

Fig. 2 Average computing times

with 4 and 5 vehicles present average gaps that are typically about 1.00% and 1.60%, respectively. As it has been said before, it is clear that H1 produces the lowest average gaps, followed by H4. Both settings are also those with higher computing times, as can be observed in Fig. 2. Nevertheless, CPU times are very reasonable: the maximum average time, 56.2 s, is reached by H1 on the 2-vehicle instances, with a maximum of 404.0 s. Figure 2 shows a curious fact: instances with more vehicles are less time consuming. We do not have a definitive explanation for it, but it could be due to the larger computing times needed by the improvement algorithms *2-interchange*, *Or-interchange*, *reversal procedure*, *change2to0*, *change1to0* and *change1to1*, embedded in the VND procedures, when are applied to longest routes.

Figure 3 depicts the number of times that each heuristic setting reaches the best known solution. The results are consistent with those presented in Fig. 1: the largest number of best solutions is achieved by setting H1, followed by H4.

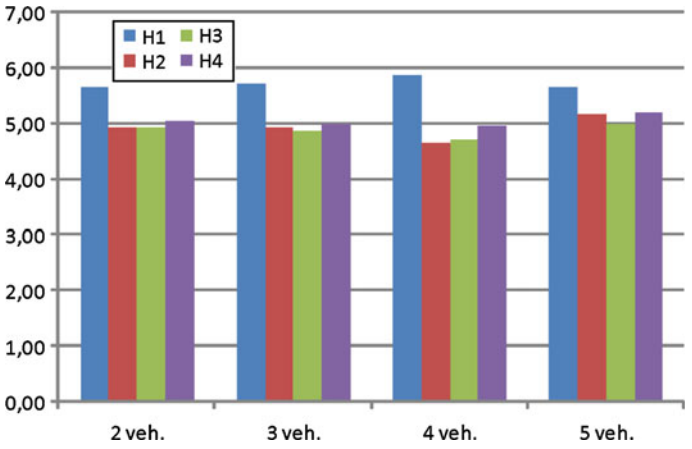

Fig. 3 Number of best solutions

Table 6 H1 versus B&C

	2 Vehicles	3 Vehicles	4 Vehicles	5 Vehicles
Number of instances	5	24	39	44
Average ratio (%)	1.73	2.61	9.17	16.22

Finally, on the instances not solved to optimality with the branch-and-cut algorithm presented in Benavent et al. (2009), we have compared the solutions obtained with version H1 of our algorithm with the best solutions found within 30 min of computation of the branch-and-cut. Table 6 shows the obtained results. Given an instance I of the MM K-WRPP, $z_{H1}(I)$ and $z_{BC}(I)$ represent the values of the solutions found by heuristic H1 and the B&C, respectively, for that instance. For each instance, the percentage ratio $(z_{BC}(I) - z_{H1}(I))/\,min(z_{H1}(I), z_{BC}(I)) * 100$ has been computed. The average values of these ratios are shown in the last row of the table. Note that all these numbers are positive, what means that the solutions obtained with H1 are up to 16.22% better on average that the best solutions found by the B&C.

Acknowledgments Enrique Benavent and Ángel Corberán thank Nicos Christofides for his help and generosity. He is a reference for many of us. Authors also wish to thank two anonymous referees for their careful reading of the manuscript and valuable suggestions, and the Ministerio de Educación y Ciencia of Spain (projects MTM2006-14961-C05-02 and MTM2009-14039-C06-02) for its support.

References

Ahr D (2004) Contributions to multiple postmen problems, PhD Thesis. University of Heidelberg, Germany, September

Ahr D, Reinelt G (2002) New heuristics and lower bounds for the min–max *k*-Chinese postman problem. In: Möring R, Raman R (eds) Algorithms-ESA 2002, 10th Annual European symposium, Rome, Italy, September 2002. Proceedings, Lecture Notes in Computer Science, vol 2461. Springer, Berlin, pp 64-74

Ahr D, Reinelt G (2006) A Tabu search algorithm for the min–max k-Chinese postman problem. Comput Oper Res 33:3403–3422
Applegate D, Cook W, Dash S, Rohe A (2002) Solution of a min–max vehicle routing problem. INFORMS J Comput 14:132–143
Assad A, Pearn WL, Golden B (1987) The capacitated Chinese postman problem: lower bounds and solvable cases. Am J Math Manag Sci 7:63–88
Barahona F, Grötschel M (1986) On the cycle polytope of a binary matroid. J Comb Theory 40:40–62
Belenguer JM, Benavent E, Labadi N, Prins C (2009) Reghioui M Split delivery capacitated arc routing problem: lower bound and metaheuristic. Submitted to Transport Sci
Benavent E, Corberán A, Piñana E, Plana I, Sanchis JM (2005) New heuristic algorithms for the windy rural postman problem. Comput Oper Res 32:3111–3128
Benavent E, Carrotta A, Corberán A, Sanchis JM, Vigo D (2007) Lower bounds and heuristics for the windy rural postman problem. Eur J Oper Res 176:855–869
Benavent E, Corberán A, Plana I, Sanchis JM (2009) Min–Max K-vehicles windy rural postman problem. Networks 54:216–226
Benavent E, Corberán A, Plana I, Sanchis JM (2010) New facets and an enhanced branch-and-cut for the min–max k-vehicles windy rural postman problem (submitted)
Christofides N, Campos V, Corberán A, Mota E (1981) An algorithm for the Rural Postman Problem, Report IC.O.R.81.5. Imperial College, London
Christofides N, Campos V, Corberán A, Mota E (1986) An algorithm for the rural postman problem on a directed graph. Math Program Study 26:155–166
Corberán A, Plana I, Sanchis JM (2008) The windy general routing polyhedron: a global view of many known arc routing polyhedra. SIAM J Discret Math 22:606–628
Corberán A, Plana I, Sanchis JM (2007) A branch & cut algorithm for the windy general routing problem and special cases. Networks 49:245–257
Delgado C, Pacheco J (2001) Minmax vehicle routing problems: application to school transport in the Province of Burgos. Lecture notes in economics and mathematical systems, vol 505, pp 297–317
Eiselt HA, Gendreau M, Laporte G (1995) Arc-routing problems, Part 2: the rural postman problem. Oper Res 43:399–414
Frederickson G, Hecht M, Kim C (1978) Approximation algorithms for some routing problems. SIAM J Comput 7:178–193
Lacomme P, Prins C, Ramdane-Chérif W (2002) Fast algorithms for general arc routing problems. IFORS, Edinburgh, 8–12 July 2002
Lacomme P, Prins C, Ramdane-Chérif W (2004) Competitive memetic algorithms for arc routing problems. Ann OR 131:159–185
Lourenço HR, Martin O, Stützle T (2002) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer's International Series, pp 321–353
Minieka E (1979) The Chinese postman problem for mixed networks. Manage Sci 25:643–648
Mladenovic N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100
Or I (1976) Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. PhD Dissertation, Northwestern University, Evanston, USA
Pearn WL (1994) Solvable cases of the k-person Chinese postman problem. Oper Res Lett 16:241–244
Raghavachari B, Veerasamy J (1998) Approximation Algorithms for the Mixed Postman Problem. In: Bixby RE, Boyd EA, Ríos-Mercado RZ (eds) Proceedings of 6th Integer programming and combinatorial optimization. LNCS, vol 1412. Springer, pp 169–179
Ramdane-Chérif W (2002) Problémes de Tournées sur Arcs. PhD thesis, University of Troyes, France [in French]
Win Z (1989) On the windy postman problem on eulerian graphs. Math Program 44:97–112