Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 160 (2005) 139–153
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Heuristic method for a mixed capacitated arc routing problem: A refuse collection application ☆

Maria Cândida Mourão *, Lígia Amado

Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa, Rua do Quelhas 6, 1200-781 Lisboa, Portugal
Available online 17 March 2004

Abstract

The capacitated arc routing problem (CARP) is known to be NP-hard. The aim of this paper is to present a new heuristic method to generate feasible solutions to an extended CARP on mixed graphs, inspired by the household refuse collection problem in Lisbon. Computational experience was done to compare the method with some well-known existing heuristics, generalised for a different extended CARP by Lacomme et al. [Fast algorithm for general arc routing problems, Presented at IFORS 2002 Conference, Edinburgh, UK], namely, the Path-Scanning, the Augment-Merge and the Ulusoy's algorithms. The results reveal a good performance of the proposed heuristic method. Generally providing a good use of the vehicles capacity, the resulting sets of feasible trips may also be considered good. The test instances involve more than 300 randomly generated test problems with dimensions of up to 400 nodes and 1220 links.

Keywords: Routing; Heuristics

1. Introduction

Designing the household refuse vehicle tours in Lisbon is usually achieved by breaking down the city in quarters. Although Lisbon has a refuse collecting fleet stationed at only one central garage, every collecting night, a fixed number of vehicles goes to a crew base within a pre-defined collecting quarter, to initiate the service. The aim of this paper is to present a new method to find high quality solutions to the optimisation of the refuse collection vehicle tours in Lisbon.

Every collecting night, each vehicle goes to the crew base in its collecting quarter, to pick up the crew. Hence, the first trip begins at the crew base, and the vehicle collects refuse along the streets, while its capacity allows it. At this point the vehicle must drive to a dumpsite, located well away from the quarter, to unload the refuse. The vehicle then goes back to the quarter and begins its second trip. Every time (except for the last) the vehicle's residual capacity so demands it must head to the dumpsite, to unload, and then return to the

☆ This paper has been partially supported by "Centro de Investigação Operacional" (FCUL)-POCTI/ISFL/152.
* Corresponding author. Tel.: +351-1-392-5868; fax: +351-1-392-2781.
E-mail addresses: cmourao@iseg.utl.pt (M.C. Mourão), lamado@iseg.utl.pt (L. Amado).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.01.023

quarter. After completing its final trip, the vehicle first goes to the crew base, drops the crew, and only then proceeds to the dumpsite to unload for the last time.

By a vehicle's *tour* we mean the total course driven since departing from the crew base point, until returning back there to drop the crew. Each tour is a sequence of two or more *trips*: the *initial trip*, from the crew base to the dumpsite, the *final trip* from the dumpsite back to the crew base and the *intermediate trips* (if any) from the dumpsite back to it.

Since the number of vehicles in the fleet is scarce for the total demand, all vehicles must be used and each vehicle performs at least two trips. The dumpsite is far away from the quarter and there is no refuse to be collected in the streets adjacent to it.

A set of tours that minimises the total collecting cost of the household refuse in the quarter may be obtained by solving a capacitated arc routing problem (CARP) with some additional constraints (see [20] for a more detailed description and for the problem formulation). The CARP is known to be an NP-hard problem [13].

As shown in two surveys [1,10,11], arc routing problems may be used to solve a huge number of real world problems in very different domains (winter gritting, postal distribution, meter reading, street sweeping, etc.). Recently, Dror [8] edited a book on arc routing, compiling many references and boarding this class of problems, as well as some related ones, from both a theoretical and a practical point of view.

The aim of this paper is to present an heuristic method that provides good feasible solutions for a mixed capacitated arc routing problem, inspired by the refuse collection problem in Lisbon. Section 2 is devoted to the main notation used herewith. Then, the graphs orientation, enabling us to fix the service direction for each narrow street that may be collected in both sides at the same time, is justified. Section 4 is devoted to the directed problem. Here some proofs on a previous presented lower bound [20] are performed and the new heuristic method is explained. Computational results for the directed case, on randomly generated data, are first presented in Section 5. The method, which can be used for the mixed case, is also tested over some benchmark problems. These problems were created by Lacomme et al. [17], [1] for their Extended CARP. With the results these authors got for some generalised well-known methods, we evaluate our new heuristic. As it can be seen, the upper bounds provided by this new heuristic are competitive.

2. Notation

Let $G = (V, A \cup E)$ be a mixed graph, with $n+1$ nodes ($|V| = n+1$), representing the quarter and the dumpsite. Node 1 represents the crew base and node $n+1$ the dumpsite. Remaining nodes represent street crossings or dead-ending streets. Each arc, $(i,j) \in A$, represents a street segment in between two street crossings. A one-way street is represented by one arc with the same orientation. Edges, $(i,j) \in E$, are used for small two-way streets, whilst the larger ones are represented by a pair of reverse arcs, $(i,j) \in A$ and $(j,i) \in A$.

We denote by

P	the number of vehicles used to collect the refuse in the quarter
W	the vehicle capacity (all the vehicles have the same capacity)
G_R	$= (V, A_R \cup E_R)$, the *demand graph*, with: $A_R \subseteq A$ the set of *demand arcs* representing one-way streets or one side of larger two-way streets requiring refuse collection; $E_R \subseteq E$ the set of *demand edges* representing two-way streets where the refuse can be collected on both sides in a single pass. Links in $(A \cup E) \setminus (A_R \cup E_R)$, if any, may be used for *deadheading*
$d^-(\ell)$	the in-degree of node $\ell \in V$ in (V, A_R), i.e. $d^-(\ell) = \lvert\{(i,\ell) : (i,\ell) \in A_R\}\rvert$
$d^+(\ell)$	the out-degree of $\ell \in V$ in (V, A_R), i.e. $d^+(\ell) = \lvert\{(\ell,j) : (\ell,j) \in A_R\}\rvert$
$D(\ell)$	$= d^-(\ell) - d^+(\ell)\ \forall \ell \in V$, the equilibrium of node ℓ in (V, A_R)
$d(\ell)$	$= \lvert\{(\ell,x) : (\ell,x) \in E\}\rvert$, the degree of node ℓ in (V, E_R).

[1] We thank Christian Prins and Wahiba Ramdane-Chérif for their collaboration.

Hereafter, a (*un*)*balanced node*, ℓ, will represent a node in the directed subgraph (V, A_R) with $D(\ell) = 0$ $(D(\ell) \neq 0)$. An unbalanced node with $D(\ell) > 0$ $(D(\ell) < 0)$ represents a *source* (*sink*). In the undirected subgraph (V, E_R), a node ℓ is classified as an *even* (*odd*) *node* if $d(\ell)$ is even (odd).

Having the arcs/edges demand and the vehicles capacity and number, the minimum number of trips needed, say L, may be easily computed.

As in [20] this problem contains all the constraints of a CARP [11] and some additional ones to define the configuration of the trips and their number. From the previous definition, initial trips correspond to P paths from node 1 to node $n+1$, final trips correspond to P paths from node $n+1$ to node 1 and intermediate trips correspond to at most $L - 2P$ circuits from node $n+1$ and back to it. Feasible solutions should then be found having these figures into account.

3. Choosing the service direction

The Service Direction algorithm takes a mixed demand graph and turns it into an oriented one, trying to maintain or improve the nodes equilibrium. This is done in order to allow its transformation into an Eulerian directed graph with a number of deadheading arcs as small as possible, as can be seen in the following section.

Corberán et al. presented methods for the rural postman problem [5] and for the mixed Chinese postman problem [6], where edges were directed one by one. Here, in each iteration, either a chain or a cycle is directed.

The algorithm begins by directing the edges linking two nodes, one of them being candidate to a source and the other to a sink. It then randomly transforms the cycles into circuits. Finally, the chains having extreme odd nodes are also oriented.

Algorithm (*Service Direction*).

0. Data: mixed graph $G_R = (V, A_R \cup E_R)$
1. Let $\overline{V} \leftarrow V$, $\overline{A}_R \leftarrow A_R$, $\overline{E}_R \leftarrow E_R$; $\overline{D}(i) \leftarrow D(i)$ and $\bar{d}(i) \leftarrow d(i)$ $\forall i \in V$.
   {Orientation of edges linking sources to sinks}
2. *For* $k = 1$ *to* $|\overline{E}_R|$
   do Let $e = (i, j) \in \overline{E}_R$ be an edge not yet analysed
   if $\overline{D}(i) \times \overline{D}(j) < 0$
   then if $\overline{D}(j) < 0$ *then* $\overline{A}_R \leftarrow \overline{A}_R \cup \{(i, j)\}$
   else $\overline{A}_R \leftarrow \overline{A}_R \cup \{(j, i)\}$;
   Update $\overline{D}(i), \overline{D}(j)$.
3. Update $\overline{E}_R$, $\bar{d}(i)$ $\forall i \in \overline{V}$ and $\overline{V}$, removing nodes with zero degree.
4. From the resulting graph, build the subgraph induced by the edges, $(\overline{V}, \overline{E}_R)$.
5. {Cycles orientation}
   While ($\exists i \in \overline{V}$: there is a cycle incident to node i)
   do Randomly direct the cycle constructing a circuit incident into i;
   Update $\bar{d}(\cdot)$ for all the nodes in the circuit;
   Update $(\overline{V}, \overline{E}_R)$ by removing the edges in the cycle and all nodes with zero degree.
   {Chains orientation}
6. *While* ($\exists i \in \overline{V} : \overline{D}(i) > 0 \wedge \bar{d}(i)$ odd)
   do Choose a maximum cardinality chain having i as extreme node;
   Direct the edges in the chain as a path beginning at node i;
   Update $\overline{D}(\cdot)$ and $\bar{d}(\cdot)$ for all nodes in the path;
   Update $(\overline{V}, \overline{E}_R)$ by removing the edges in the chain and all nodes with zero degree.
7. *While* ($\exists i \in \overline{V} : \overline{D}(i) < 0 \wedge \bar{d}(i)$ odd)
   do Choose a maximum cardinality chain having i as extreme node;
   Direct the edges in the chain as a path ending at node i;
   Update $\overline{D}(\cdot)$ and $\bar{d}(\cdot)$ for all nodes in the path;
   Update $(\overline{V}, \overline{E}_R)$ by removing the edges in the chain and all nodes with zero degree.
8. *While* ($\exists i \in \overline{V} : \overline{D}(i) = 0 \wedge \bar{d}(i)$ odd)
   do Choose a maximum cardinality chain having i as extreme node; let j be the other one;
   Direct the edges in the chain as a path with the initial node randomly chosen between i and j;
   Update $\overline{D}(\cdot)$ and $\bar{d}(\cdot)$ for all nodes in the path;

Update $(\overline{V}, \overline{E}_R)$ by removing the edges in the chain and all nodes with zero degree.

The following example illustrates the algorithm.

Example. Let us consider the mixed demand graph in Fig. 1, where numbers next to the nodes are equal to $D(\cdot)$, positive numbers representing nodes supply, whilst negative numbers represent nodes demand.

In step 2, the edge $(5,6)$ is directed from the source node 5 to the destination node 6, resulting in the arc $(5,6)$. After the cycles orientation (step 5), and assuming that the chosen direction was $(3,10,9,8,3)$, the graph induced by the edges, is shown in Fig. 2.

Next, at step 6, a maximum cardinality chain having the source 11 as extreme node is identified and directed from 11. Then, the path $(11,4,2)$ is removed from the graph, resulting the subgraph in Fig. 3.

At step 7, the chain into the sink node 1, is directed as the path $(3,1)$. Finally, at step 8, the remaining chain between balanced and odd nodes

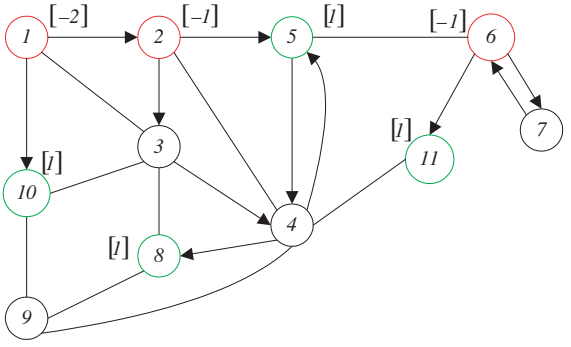

Fig. 1. Mixed demand graph.

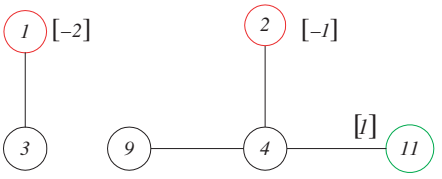

Fig. 2. Subgraph induced by the edges subset at the end of step 5.

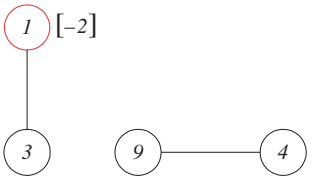

Fig. 3. Subgraph induced by the edges subset at the end of step 6.

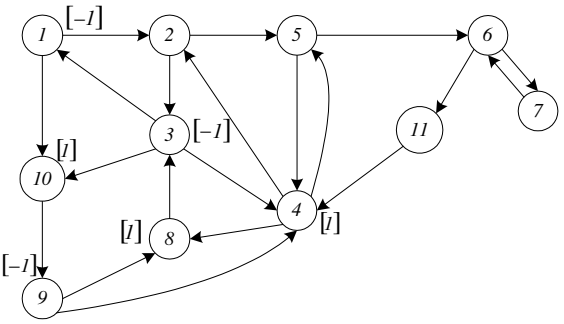

Fig. 4. Directed demand graph solution of the Service Direction algorithm.

is randomly directed, say as $(9,4)$. The directed graph now becomes the one plotted in Fig. 4, with the total demand (and supply) decreased by one.

Let us now show the validity of the algorithm, namely that with this method any mixed graph may be transformed into a directed one. With this purpose, one first proves two simple lemmas.

Define a *chain* (*path*) *inner node* as a node in the chain (path) different from its extreme nodes.

Lemma 1. *Let $G = (V, E)$ be a general undirected graph with no cycles. If a node $i \in V$ is even ($d(i)$ even), then there is a chain between two odd nodes having i as one of its inner nodes.*

Proof. Consider a maximum cardinality chain, C1, having i as one of its extreme nodes. Let j be the other extreme node. Since C1 has maximum cardinality then $d(j) = 1$, given that otherwise the chain could be augmented with, at least, one more edge (and forming no cycles).

With i being an even node, there is another maximum cardinality chain, C2, with i as one of its

extreme nodes and having no inner point in common with C1 ((V,E) is acyclic). Let k be the other extreme node of C2. Then $d(k)=1$ and there is a chain, $C=\mathrm{C1}\cup\mathrm{C2}$, with extreme odd nodes (j and k) and having i as one of its inner nodes. □

Lemma 2. *At the end of step 7 of Service Direction algorithm, for all node i, one of the following conditions is verified*:

(*a*) $\overline{D}(i)=0\wedge\bar{d}(i)$ *odd*,
(*b*) $\bar{d}(i)$ *is even*.

Proof. After each iteration of step 6, a node $i\in\overline{V}:(\overline{D}(i)>0\wedge\bar{d}(i)$ odd) which was chosen as an extreme point of a chain, becomes an even node, whilst for the other extreme node, say j, $\bar{d}(j)=0$ (Lemma 1). All the other inner nodes do not change their equilibrium, $\overline{D}(\cdot)$, nor their parity ($\bar{d}(\cdot)$ is reduced by 2). Thus, during step 6 no new odd nodes are created. Subsequently, at the end of step 6, no new candidates for the following steps are created and

$$\{i\in\overline{V}:\overline{D}(i)>0\wedge\bar{d}(i)\mathrm{odd}\}=\emptyset.$$

Similarly, during step 7, no new candidates for steps 8 or 6 appear and

$$\{i\in\overline{V}:\overline{D}(i)<0\wedge\bar{d}(i)\mathrm{odd}\}=\emptyset.$$

So, at the end of step 7, for all node i, $(\overline{D}(i)=0\wedge\bar{d}(i)$ odd) or $\bar{d}(i)$ is even. □

It is now easy to prove that with the proposed algorithm all the edges in the initial mixed graph are directed.

Proposition 1. $\overline{E}_R=\emptyset$ *at the end of the Service Direction algorithm.*

Proof. Consider the subgraph $(\overline{V},\overline{E}_R)$ obtained at the end of step 7.

If there is any odd node $i\in\overline{V}$, then $\overline{D}(i)=0$ (Lemma 2). In one of the iterations of step 8 node i is then taken as an extreme node of a chain, one of its incident edges is directed, and i becomes an even node or $\bar{d}(i)=0$.

If $i\in\overline{V}$ is even, then by Lemma 1, i is inner to a chain with two extreme odd nodes, say j and k. At the end of step 7, these two odd nodes have $\overline{D}(j)=\overline{D}(k)=0$ (Lemma 2). So, during step 8, node i remains even until $\bar{d}(i)=0$, i.e. until the edges incident in i become arcs.

Thus, $\overline{E}_R=\emptyset$ at the end of step 8 of the Service Direction algorithm. □

Let us now highlight some features of the node degrees, which will be useful later on.

Remark 1. In any undirected graph, it turns out obvious, that a node i with degree $d(i)\geqslant 2$ has $\lfloor d(i)/2\rfloor$ disjoint chains or cycles with i as one of its inner nodes. Moreover, if $d(i)$ is odd, then $\lfloor d(i)/2\rfloor=(d(i)-1)/2$. So, the remaining chain incident in i has i as one of its extreme nodes.

Next, it will be seen the usefulness of the directed demand graph obtained by the Service Direction algorithm. From now on it will be denoted by SD graph.

4. Bounds for the directed problem

In this section, the graph is the directed one, SD, and the problem consists in finding feasible vehicle tours for the set of P vehicles on this directed graph.

Next a lower bound is computed from the directed network, where data for a new arc in the set $\overline{A}_R\setminus A_R$, corresponds to its initial edge data. From the multigraph obtained with this procedure an heuristic method, HM, is then justified. The HM provides feasible solutions both for the directed and for the mixed problems.

4.1. Lower bound

One must note that an Eulerian network may be obtained from a directed one, if a transportation problem (TP) is defined for the subset of unbalanced nodes [3]. Sources are unbalanced nodes, with the in-degree greater than the corresponding out-degree, and sinks represent unbalanced nodes with the out-degree greater than the in-degree, as illustrated next. The dumpsite represents both a source and a sink, with supply and demand given

by the total graph demand and the vehicles capacity and number. Such TP is balanced and its solution gives a valid lower bound for our directed problem [20].

Example. Consider the directed graph from the previous section (Fig. 4) with three sources and three sinks, with unitary demands and supplies. Assuming that one vehicle with capacity 7 is available, and that all the edges demand are equal to one (so the total demand is 20), the minimum number of arcs incident into the dumpsite (node $n+1=12$) is given by $\lceil \frac{20}{7} \rceil - 1 = 2$. To get a balanced graph a transportation problem is solved, where the unit transportation cost from each source to each sink is given by the shortest deadheading path cost between the corresponding nodes. The arcs in the transportation solution (plotted in Fig. 5 as dashed arcs) represent paths that must be deadhead to obtain an Eulerian graph and to ensure a minimum number of arcs incident into the dumpsite.

In this section it is proved that the resulting Eulerian graph has a small number of deadheading arcs, where both feasible and good solutions may be found.

The following proposition gives us an upper bound on the total number of deadheading arcs resulting from the TP solution, by fixing an upper bound on the total TP demand and supply. Note that, the value of a TP variable indicates the number of deadheading arcs that should be added between the corresponding source and sink.

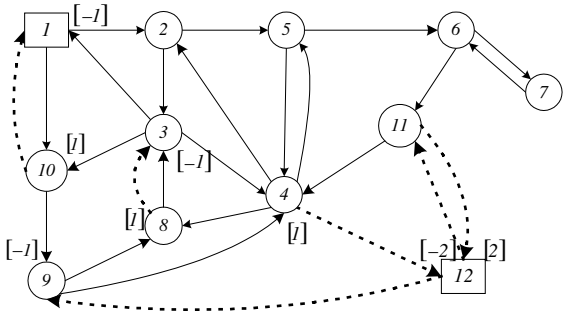

Fig. 5. Directed Eulerian multigraph.

Proposition 2. *Let* $G=(V, A \cup E)$ *be the initial mixed network and define*:

s^0 *as the total TP supply in the initial mixed network*,
s^1 *as the total TP supply in the directed SD network*,
$Y = \{(i,j) \in E : D(i) \times D(j) < 0\}$,
$B = \{i \in V : D(i) = 0 \wedge d(i)$ *is odd*$\}$.

Then

$$s^1 \leqslant s^0 - |Y| - \begin{pmatrix} \textit{number of chains between} \\ \textit{odd sources and odd sinks} \end{pmatrix} + \frac{|B|}{2}.$$

Proof. It is obvious that $s^0 - |Y|$ represents the total TP demand at the end of step 2.

When chains and cycles are directed, the inner nodes classification remains unchanged. In fact, for each inner node, two edges are directed, one into the node and the other out of the node. So, a (un)balanced inner node remains (un)balanced, as initially and does not affect the total TP demand.

Hence, after applying step 5, the total TP demand remains $s^0 - |Y|$.

As justified before (Remark 1), an odd node i with degree $d(i)$, is an extreme of a chain and inner to $(d(i)-1)/2$ chains or cycles. Thus, the extreme nodes are the ones that may change its initial classification, and conduct to a different TP total demand.

So, lets look into the chains with two odd nodes as extreme nodes: i and j. The following cases should be considered:

(A) $\overline{D}(i) > 0$, so i is a source with supply $\overline{D}(i)$.
At step 6, the chain is directed from i and $\overline{D}(i) = \bar{d}^-(i) - (\bar{d}^+(i)+1) \geqslant 0$.
(A.i) $\overline{D}(j) \geqslant 0$, so j is a source with supply $\overline{D}(j)$.
The chain is directed to j, so $\overline{D}(j) = (\bar{d}^-(j)+1) - \bar{d}^+(j) > 0$. Then, the total TP demand and supply remains unchanged, but note that one unit of supply may be transferred from i to j.

(A.ii) $\overline{D}(j) < 0$, so j is a sink with demand $(-\overline{D}(j))$.
The chain is directed to j, so $\overline{D}(j) = (\bar{d}^-(j) + 1) - \bar{d}^+(j) \leqslant 0$. Then, the total TP supply decreases one unit as does its total demand.

(B) $\overline{D}(i) < 0$, so i is a sink with demand $(-\overline{D}(i))$.
At step 7, the chain is directed into i, and $\overline{D}(i) = (\bar{d}^-(i) + 1) - \bar{d}^+(i) \leqslant 0$.
At this step, $\overline{D}(j) \leqslant 0$, so j is a sink with demand $(-\overline{D}(j))$, once other cases where considered by step 6.
Since the chain is directed from j, $\overline{D}(j) = \bar{d}^-(j) - (\bar{d}^+(j) + 1) < 0$. Then, the total TP demand and supply remains the same.
So, after steps 6 and 7, the total TP demand decreases one unit for each chain directed from a source to a sink.

(C) $\overline{D}(i) = 0$ and $\overline{D}(j) = 0$, so i and j are balanced nodes.
At step 8, if the chain is directed from i to j one gets

$$\overline{D}(i) = \bar{d}^-(i) - (\bar{d}^+(i) + 1) < 0 \quad \text{and}$$
$$\overline{D}(j) = (\bar{d}^-(j) + 1) - \bar{d}^+(j) > 0.$$

Then, both total TP demand and supply increase one unit. The same applies for the chains directed from j to i, obviously.
As $|B|$ represents the number of balanced and odd nodes, the maximum number of chains between balanced nodes identified at step 8 is no greater than $|B|/2$. In fact, some of these nodes may be previously used as end nodes of paths from sinks or sources. □

From our computational experience, we were able to establish that, generally, $|B|$ is a small number, conducting to a balanced TP with small total demand, which implies that a small number of deadheading arcs are needed to obtain an Eulerian network.

Similarly to the TP presented in [20], the TP here defined also considers the dumpsite both as a source and a sink. This is also the case for the crew base point in some instances. In doing so the resulting directed problem is similar to the one defined before, and so is the lower bound proof.

Proposition 3. *Let C_S be the total service cost, i.e., the sum of the service cost of all link (arc and edges) demand, z_{TP} be the optimal value of the TP, and z^* be the optimal value for the refuse problem over the directed graph. Then $z_{LB} = C_S + z_{TP} \leqslant z^*$.*

4.2. Heuristic method

4.2.1. Introduction

In trying to get similar feasible vehicle tours, with a few number of intersections, an heuristic method was developed, based on the Eulerian and directed network previously built.

First, a set of small vehicle trips is identified, representing trips with a total amount of refuse much lower than the vehicles capacity. Then, the trips aggregation is allowed between trips that intersect each other and if the total resulting trip demand does not exceed the vehicles capacity. For this purpose, a trips multigraph is defined and pairs of trips are joined together, based on a matching solution process. Finally a feasible solution is built from the feasible vehicle trips generated. The following diagram generally describes the algorithm (Diagram 1).

At this point the main steps of this heuristic are presented. Firstly, a procedure for the small trips identification is described. Secondly the trips multigraph generation is discussed. Then a method based on a matching procedure for the trips aggregation is justified. At the end of each iteration, the trips multigraph is updated and the aggregation phase ends when the set of edges of the trips multigraph is empty, i.e., when no more aggregations are allowed. Finally a feasible solution is achieved, after comparing the number of initial and final trips with the vehicles number.

4.2.2. The trips identification method

The aim of this initial phase is to get a set of small trips, with respect to the total demand, i.e., the total amount of refuse included per trip. In some heuristic methods proposed, with the same imbedded perspective, the authors assume that each initial circuit has only one demand arc (see [13,14]). Other kind of initial circuits where also suggested in [4,9,15,16,19,21].

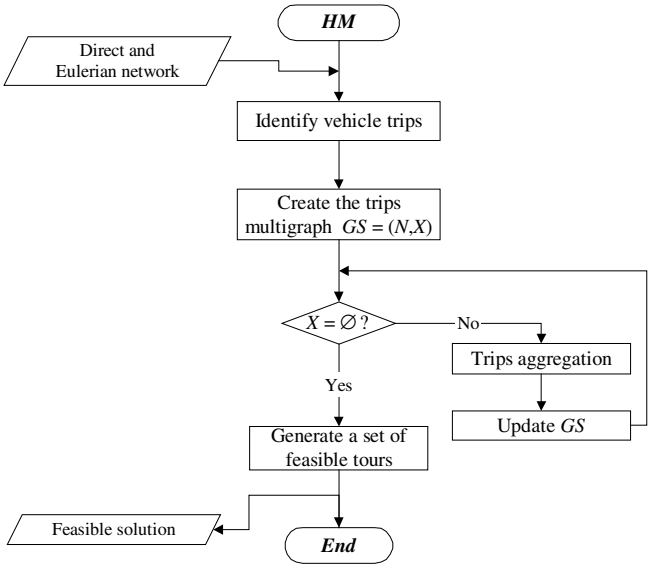

Diagram 1. Heuristic method.

Here, based on the directed and Eulerian multigraph previously constructed, the method starts by identifying the minimum demand circuit incident to each node. From these circuits the biggest one is chosen. In doing so, one tries to get a set of small and similar trips, in respect to the total demand included per trip.

From the selected circuit, feasible vehicle trips are then built. For this purpose, when the trips include the crew base node, 1, initial and/or final trips are created, depending on the demand arcs incidents into 1. In all other cases, intermediate trips are built. The trips are linked to the dumpsite from the nearest point of the circuit in study. In building the trips one has also to consider if the dumpsite itself belongs to the identified circuit, since in these cases no further links to the dumpsite are needed.

The arcs in the circuit are now removed from the Eulerian graph and the method is repeated until all demand arcs belong to a feasible trip.

Note that all demand arcs which are replicated as deadheading arcs, should be, first of all, considered as demand arcs, as shown in the following example.

Example. Consider once more the Eulerian and directed graph from the previous section (Fig. 5) without the deadheading circuits incident into the dumpsite, with unitary demands for all demand arcs. The demand of the minimum demand circuit incident to each node is represented next to each node in Fig. 6.

So, the first circuit selected and removed (the one with the greater total demand) is the circuit incident into node 11, leading to the following feasible intermediate trip: $T1 = (12, 11, 4, 5, 6, 11, 12)$. Next, the circuit incident into 2 is chosen and removed, $C2 = (2, 3, 1, 2)$. As its incident into the special node 1, the circuit is broken down in two feasible trips, a final trip, $T2 = (12, 3, 1)$ and

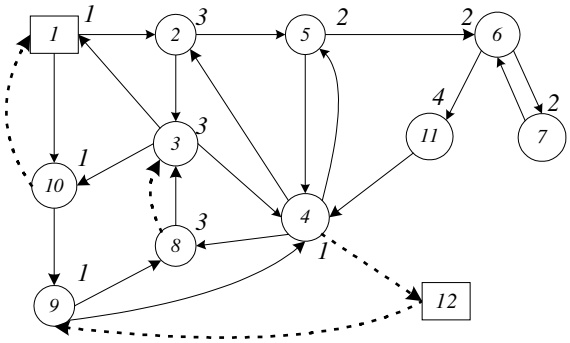

Fig. 6. Shortest demand circuits identification.

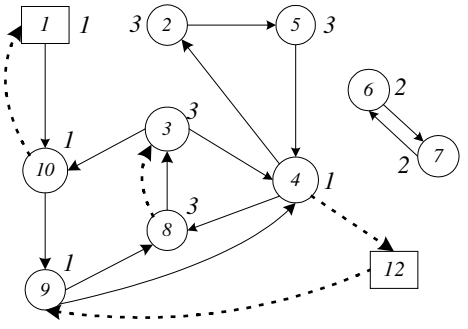

Fig. 7. Shortest demand circuits identification.

an initial one, $T3 = (1, 2, 3, 12)$, and the graph becomes the one in Fig. 7.

Another circuit incident into 2 is then chosen and transformed into the feasible intermediate trip $T4 = (12, 4, 2, 5, 4, 12)$. The method proceeds until no more demand arcs exist in the remaining graph. In this case, the next feasible trips are: $T5 = (12, 4, 8, 3, 4, 12)$, generated from the circuit incident into node 3; $T6 = (12, 3, 10, 9, 8, 12)$, from another circuit incident into 3; $T7 = (12, 6, 7, 6, 12)$, from the circuit incident into 6; $T8 = (1, 10, 12)$, from the circuit incident into 1; $T9 = (12, 9, 4, 12)$, from the circuit incident into 4.

4.2.3. The trips multigraph

In this second phase, the goal is to generate a trips multigraph where it is possible to easily identify the best aggregation. So, a multigraph (GS) is built, where each trip is represented by a node. An edge exists whenever the corresponding trips are adjacent and its fusion is compatible with the vehicles capacity. Throughout this paper it is assumed that *two trips are adjacent* if they have, at least, one service node in common, i.e., an extreme of one or more demand arc or edge. If two trips intersect in two service nodes, two edges are considered between the respective nodes, in the trips multigraph. The cost of an edge represents the saving of the corresponding trips aggregation. This cost depends on the intersection node. In fact, different costs may be computed for the same two trips fusion if they have more than one intersection node.

Note that the aggregation of two intermediate trips implies the elimination of two links between the demand graph and the dumpsite, whilst with other trips fusion only one of such links is removed. Furthermore, it isn't allowed the fusion between two initial or final trips, neither between an initial and a final trip, until the last step of the heuristic method. Therefore in the first trips multigraph there are no edges between nodes representing initial and/or final trips.

In sum, a trips multigraph is built, where nodes represent trips and edges represent feasible trips aggregation. Male and Liebman [19] were the first authors to propose a cycles network for a similar problem. The more relevant difference towards the here presented method is depicted in the cycle's identification process described in the previous section.

Example. With one vehicle with capacity 7, the trips multigraph for the previous circuits set is represented in Fig. 8.

Note that as trips T1 and T4 (T5 and T6) have two nodes in common, two edges between the corresponding nodes are considered in the trips multigraph.

4.2.4. Trips aggregation

At each iteration, the pairs of trips to be joined together are chosen according to a matching solution in the trips multigraph, GS, where one seeks to maximise the total savings.

With this method, from the aggregation of two intermediate trips an intermediate one is built,

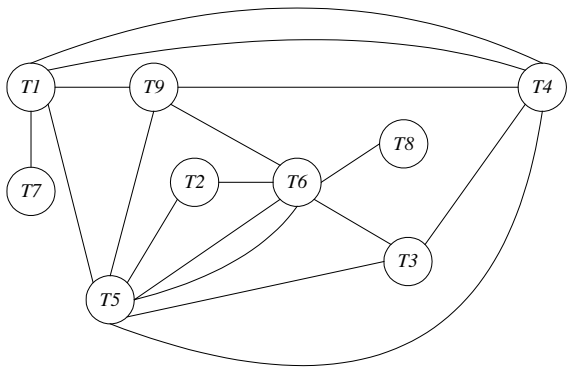

Fig. 8. Trips multigraph, GS.

whilst from the aggregation of an initial (final) trip and an intermediate one, we get an initial (final) trip. This is imposed because the number of initial/final trips is fixed so as to obtain a feasible set of vehicle tours.

From a maximum savings matching solution, it is possible to get the best set of fusions between pairs of trips, in each iteration. Since the number of trips could be odd, there may not exist a maximum perfect matching solution. Notwithstanding the matching algorithms being polynomial, it is known that they also are time consuming. On the other hand, savings maximisation, at each iteration, does not guarantee the best final solution. These facts justify the option of finding feasible matching solutions, instead of seeking the optimal matching one.

In short, the aggregation phase is based on a feasible solution to the matching problem over the trips multigraph, GS. At each iteration, $\lfloor N/2 \rfloor$ edges are heuristically chosen to be the matching solution ones, as stated next:

(i) Sort the edges in decreasing order of their savings. Equal savings are resorted by the edge:
   (a) with more replicas, as it corresponds to a pair of trips with more than one intersection point;
   (b) incident to a node with smaller degree, since it represents a trip with fewer chances of being aggregate;
   (c) representing an aggregation between the farthest trips from the dumpsite.
(ii) Starting from the first element of the sorted list, select the maximum number of edges that may represent a matching feasible solution. These edges represent the trips that should be aggregated.

This process is repeated after the corresponding trips are patched together, and the trips multigraph is updated, until it has no more edges, i.e. no more trips can be aggregated.

Example. If the edges in the matching solution, for the multigraph in Fig. 8, are (T1, T7), (T2, T5), (T4, T9) and (T6, T8), the new trips are: $T1 = (12, 11, 4, 5, 6, 7, 6, 11, 12)$, $T2 = (12, 3, 4, 8, 3, 1)$, $T3 = (1, 2, 3, 12)$, $T4 = (12, 9, 4, 2, 5, 4, 12)$ and $T5 = (1, 10, 9, 8, 3, 10, 12)$. With one vehicle with capacity 7, and if final and initial trips can only be aggregated with intermediate ones, the next fusion must be between trips T3 and T4, resulting in the new trip $T3 = (1, 2, 5, 4, 9, 4, 2, 3, 12)$. No more aggregations are allowed, considering to the capacity and the trips total demands, and the fusion phase ends.

4.2.5. Feasible solution

After the aggregation phase, a set of feasible trips is achieved. Having a fixed number of vehicles, this set represents a feasible solution if the number of initial and final trips equals the vehicles number, P. So, to obtain the final feasible solution, the number of existing initial and final trips is compared with P and, if needed, some starting/ending points are then changed, or some aggregations between initial and/or final trips are allowed.

Hence, this heuristic method produces a feasible solution on a directed graph. As we are able to fix the service direction for each demand edge, representing two-way streets that may be collected in a single pass, this method may, obviously, be applied to the mixed problem. The feasible solution value represents an upper bound on the optimum value for the refuse problem for any refuse graph.

Example. Existing only one vehicle, one of the two initial trips generated must turn into an intermediate trip. Suppose that choosing trip T5 represents the more cost effective choice. Hence, the feasible solution is given by the initial trip $T3 = (1, 2, 5, 4, 9, 4, 2, 3, 12)$, the two intermediate trips $T1 = (12, 11, 4, 5, 6, 7, 6, 11, 12)$, and $T5 = (12, 1, 10, 9, 8, 3, 10, 12)$, and the final trip $T2 = (12, 3, 4, 8, 3, 1)$.

5. Computational experience

We briefly report the computational experience made so as to evaluate the quality of the feasible solutions obtained by the heuristic method here presented, HM.

All programs were implemented in Delphi 5.0, and executed on an Intel Pentium II running at 300 MHz, with 64 MB RAM.

As the lower bound validity was proven only for directed networks, the gap values, using this lower bound, may only be computed for directed problems. So, the mixed randomly generated graphs are directed, and the first data set tested was generated to allow an evaluation of the heuristic performance on real simulated data (with the dumpsite far away from the refuse quarter). In the second randomly generated data set it is assumed that a dumpsite or a transfer station exists inside the refuse quarter. The method is compared with some well-known methods at the end of this section.

5.1. Computational results for the directed problem

First let us discuss the results obtained on randomly generated networks similar to the Lisbon city network (see [20]), as we vary its dimensions.

Table 1 depicts the results for the first data set. Each row in this table shows the average values of ten similar test problems (hence 260 instances were randomly generated). The first column shows a number of nodes varying between 65 and 401, including the dumpsite. As in the real application, the number of vehicles was set equal to 2. For each number of nodes, different number of links (edges, arcs, deadheading or demand—second column) was tried, leading to different number of demand arcs and edges (third and fourth columns). This number varies between 66% and 72% of the total links. The graphs are sparse, as usually the city networks are, having no more than 1220 links.

For each dimension, vehicles with different capacities were also considered (fifth column), along with diverse arcs and edges demand, which lead to various total demands (sixth column).

As usual, LB and UB denote lower and upper bound values for the optimum, respectively, and the gap is computed by $\text{gap} = (\text{UB} - \text{LB})/\text{UB} \times 100$. For each number of nodes, the gap column accentuates in bold the minimum gap value, whilst the maximum gap value is shown in italic.

A very interesting result for real world problems lies in the fact that the vehicles carry in excess of 70% of their full capacity in the majority of trips (see columns 10–12), in spite of the less encouraging gap values.

The caption in column 10 refers to the number of trips where the vehicles carry less than 50% of their full capacity. In fact, the vehicles always travelled with a 30% load, at least, except in one out of the 260 randomly generated problems.

When the dumpsite is far away from the quarter gap values are usually biased by the difference between the minimum number of trips—given by the lower bounding algorithm—and the number of trips found by the feasible solution procedure; as a consequence, another set of randomly generated test problems was tried assuming that the dumpsite location is within the refuse quarter. The results are depicted in Table 2. Each row in this table contains the mean value of five test problems. In this case the difference between the minimum number of trips and the actual number of trips do not strongly affect the gap values, since arc distances are all within the same range of values. Although this is not the real case in Lisbon, these problems yield a set of significantly smaller gap values, concurring to the idea that gap values are prejudiced by the lower bound value. In general, test problems found in literature fall within this scope (see, for instance, [5,7,12]). This paper addresses problems with significantly higher dimensions (from 65 to 401 nodes), with gap values varying between 0.8% and 3% and small execution times (from 0 seconds to 12 minutes).

5.2. Computational results for the mixed problem

As referred, if the SD graph is identified by the Service Direction algorithm, and set as the input graph for the HM presented in Section 4, feasible solutions for the mixed problem may be computed.

Lacomme et al. [18] developed powerful *memetic* algorithms for their extended CARP (ECARP), where they were able to deal with some real situations. In order to test its quality they needed a set of good initial feasible solutions for the problem. With this purpose, they extended some well-known methods, namely, the Augment-Merge

Table 1
Computational results for randomly generated problems

Nodes	Links	Demand		W	Total demand	LB	UB	gap (%)	% of trips occupying			Time (second)
		Arcs	Edges						<50%W	50–70%W	>70%W	
65	144	72	22	1410	5454	8715	10370	16.0	0	1	4	
	148	80	22	1518	6209	9270	11246	17.6	1	1	3	
	160	79	28	1196	6634	11968	14565	17.8	0	2	5	
	192	84	47	2233	8211	8934	10707	16.6	0	2	3	
	188	96	39	1807	8366	10641	13250	*19.7*	1	1	5	
	188	92	43	1563	8297	12520	14376	**12.9**	1	1	5	
Mean	170	84	34	1621	7195	10341	12419	16.7	0	1	4	0.5
170	400	206	64	4010	16194	24141	29782	18.9	0	1	4	
	408	204	68	3842	16494	24057	31261	23.0	0	2	4	
	413	205	71	3018	16757	31223	36304	14.0	0	2	5	
	492	235	112	5598	21170	22498	28543	21.2	0	1	3	
	506	238	118	4813	21734	27616	30174	**8.5**	0	1	4	
	500	241	114	4076	21578	32817	43515	*24.6*	1	4	3	
Mean	453	222	91	4226	18988	27059	33263	18.7	0	2	4	18.2
257	609	314	96	6070	24548	37070	41666	**11.0**	0	1	4	
	624	308	107	4480	25214	47098	60276	21.9	1	2	5	
	612	312	97	3950	24788	54468	66613	18.2	1	2	5	
	750	356	176	8547	32103	33906	44967	*24.6*	1	1	4	
	761	359	178	7230	32564	41658	48660	14.4	1	1	5	
	757	359	177	5400	32497	52847	67685	21.9	1	3	5	
Mean	686	335	139	5946	28619	44508	54978	19.0	1	1	5	115.3
401	959	494	157	9708	39102	57049	68259	16.4	0	2	3	
	986	483	175	10022	39611	54009	65976	18.1	0	1	3	
	981	484	166	6126	39025	85470	99728	14.3	0	2	6	
	971	482	158	4793	38433	103218	130004	20.6	1	3	7	
	1178	551	277	13373	50169	52954	61731	**14.2**	1	0	4	
	1189	547	284	11250	50227	64273	81454	21.1	1	3	3	
	1215	561	288	8607	51446	81072	102904	*21.2*	1	2	5	
	1185	554	280	6743	50494	99931	117895	15.2	1	1	7	
Mean	1083	520	223	8828	44813	74747	90994	17.9	0	2	5	721.0

Table 2
Computational results for problems with the dumpsite located inside the quarter

Nodes	Links	Demand		W	Total demand	LB	UB	gap (%)	Time (second)
		Arcs	Edges						
65	193	91	44	1410	8473	4759	4921	3.30	0.5
122	350	155	87	2506	14764	8556	8726	1.96	5.0
145	437	218	95	3114	19050	10884	11168	2.54	10.8
170	498	246	111	3566	23642	12562	12745	1.43	19.4
197	586	266	144	4227	24963	14411	14755	2.34	33.7
226	671	323	154	4806	29019	16627	16862	1.38	61.6
257	753	350	178	5378	32324	15969	16253	1.75	116.8
290	857	411	194	6070	36710	21120	21517	1.85	168.7
325	965	466	220	6860	41515	24106	24668	2.27	288.0
362	1092	530	249	7773	47070	27259	27480	0.80	505.7
401	1189	566	276	8447	50883	29420	29798	1.27	719.7

[13], the Path-Scanning [14,15], and the Ulusoy's [22]. The authors also extended the method proposed by Golden and Wong [13], without the augment phase, resulting into the Extended Merge algorithm (EM). As they claim, this usually provides better bounds than the first one, with the two phases.

The LPR data problems, created by Lacomme et al. [17] were used to compare our heuristic with the extended methods referred to above. These problems were built for a different ECARP. They study the refuse collection in France, were they usually have the dumpsite inside the quarter, whilst we assume to have two special nodes, the crew base, inside the quarter, and the dumpsite very far away from the quarter. The results they got, and to which they gave us access to, are depicted from columns 7 to 10 in Table 3. The first five columns include the problem name and the respective dimensions. Lower bound values, obtained by Belenguer et al. [2], are in the sixth column.

The LPR problems were divided into three classes. As may be seen, the B problems are sparser. The A and the C problems have the same kind of density, but the last ones have a greater number of edges.

The UB1 column reflects the upper bound that was computed with HM, assuming only one collecting vehicle. Then, based on the number of trips in the feasible solution (12th column), the vehicles number was increased, if possible, and the HM produces a new upper bound, UB2 (13th column). Note that the increased number of vehicles should not exceed half of the trips number in the first solution, since each vehicle performs, at least, two trips (the initial and the final ones). This may be done to compare all the solution values, since the two extended CARPs have differences, such as for instance, the dumpsite location.

In Table 3, one can see that the upper bound computed with HM may be considered similar to the other bounds, in the sense that it outperforms 4 out of the 15 benchmark problems, also for four times the second best value is improved and only once get the worst bound.

Although these tables do not show the trips type, they were like the ones found for the first class of problems, generally occupying more than

Table 3
Computational results for LPR problems comparing our heuristic (UB1, UB2) with extended Path-Scanning (EPS), extended Ulusoys (EUL), extended Augment-Merge (EAM) and extended Merge (EM)

Problem	Nodes	Links	Demand		LB	EPS	EUL	EAM	EM	UB1	Trips Nr	UB2	Time (second)
			Arcs	Edges									
LPR-A-01	29	95	52	0	13484	13601	*13537*	13670	**13504**	***13522***	2	***13522***	0*
LPR-A-02	54	175	104	5	28052	29228	***28580***	28960	*28704*	**28473**	3	**28473**	0.3
LPR-A-03	147	503	304	33	76115	79285	*78151*	78431	**77280**	***77628***	8	***77628***	11
LPR-A-04	196	686	503	34	126946	132949	131874	**130449**	***130718***	132040	14	*131840*	45
LPR-A-05	322	1115	806	58	202748	215416	*212167*	**209724**	***210890***	212864	20	212777	387
LPR-B-01	29	69	50	5	14835	14952	**14868**	14944	***14877***	14969	2	*14933*	0.1
LPR-B-02	54	127	101	9	28654	29505	*28947*	29044	29224	***28911***	3	**28870**	0.3
LPR-B-03	164	388	305	26	77859	80558	*79910*	79972	***79808***	**78818**	8	**78818**	26
LPR-B-04	249	591	501	8	126932	133530	*132241*	**130447**	***130898***	134278	15	134278	76
LPR-B-05	402	914	801	37	209805	223535	*219702*	**216276**	***218375***	220190	22	220051	470
LPR-C-01	29	92	50	39	18639	18874	**18706**	19038	*18855*	***18719***	2	***18719***	0*
LPR-C-02	54	171	100	77	36339	*36870*	***36760***	37466	37042	37095	5	**36737**	0.2
LPR-C-03	164	558	302	241	111124	115421	***114493***	115242	**114354**	116361	12	*115139*	11
LPR-C-04	278	967	504	362	168441	174655	*173153*	175207	**171583**	173922	20	***173010***	91
LPR-C-05	370	1229	803	387	257911	268125	***266001***	*266131*	**263472**	267236	29	266473	408

* If the execution time was less than 0.05 second.

70% of the vehicles capacity. We were able to get good feasible solutions, with gap values between 0.28% and 5.47%, in short computation times (from 0 seconds to 8 minutes).

6. Final remarks

We got good bounds for the LPR data problems, showing that our heuristic is competitive with the EPS, EU, EAM and EM methods.

The HM was also analysed with a set of randomly generated problems, in terms of assessing the quality of the gap values. Results were not encouraging for a first data set, the one most closely representing the Lisbon case on analysis. Still it must be noted that these gaps were measured from the lower bounds regardless of the optimum values. Whenever the dumpsite was assumed to belong to the refuse quarter, the gaps decreased, assuming values between 0.8% and 3%.

On the other hand, the resulting sets of trips seem to represent good ones, in the sense that the vehicles capacity is properly used. We believe that the homogeneity of trips is due to the aggregation phase based on matching solutions, and to the fact that we start with a set of small demand circuits. Based on the same trips graph, some new methods are under study.

All the results were obtained with small computation times. For the largest instances (401 nodes) feasible solutions and upper bound values are computed in no longer than 12 minutes.

Some effort needs to be put in trying to devise a better lower bound method, one which may represent a more favourable valid bound to our mixed problem.

We are currently studying the inclusion of badturns restrictions in the presented methods. In the attempt to include these fundamental conditions (more realistic) we aim to preserve the data structure since failure to do so may contribute to increase the data dimensions.

References

[1] A.A. Assad, B.L. Golden, Arc routing methods and applications, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), Handbooks in Operations Research and Management Science, Network Routing, vol. 8, Elsevier Science Publishers, Amsterdam, 1995, pp. 375–483.

[2] J.M. Belenguer, E. Benavent, P. Lacomme, C. Prins, Heuristics and lower bounds for the mixed capacitated arc routing problem, Working Paper, Univ. Tecnologie de Troyes, France, 2003.

[3] E.J. Beltrami, L.D. Bodin, Networks and vehicle routing for municipal waste collection, Networks 4 (1974) 65–94.

[4] E. Benavent, V. Campos, A. Corberán, E. Mota, The capacitated arc routing problem. A heuristic algorithm, Qüestiió 14 (1990) 107–122.

[5] A. Corberán, R. Martí, A. Romero, Heuristics for the mixed rural postman problem, Computers & Operations Research 27 (2000) 183–203.

[6] A. Corberán, R. Martí, J.M. Sanchis, Approximate solutions for the CPP on a mixed graph, in: TRISTAN IV, The Triennial Symposium on Transportation Analysis, Açores, Portugal, 2001.

[7] A. Corberán, R. Martí, E. Martínez, D. Soler, The rural postman problem on mixed graphs with turn penalties, Computers & Operations Research 29 (2002) 887–903.

[8] M. Dror (Ed.), Arc Routing: Theory, Solutions and Applications, Kluwer Academic Publishers, 2000.

[9] R.W. Eglese, Routeing winter gritting vehicles, Discrete Applied Mathematics 48 (1994) 231–244.

[10] H.A. Eiselt, M. Gendreau, G. Laporte, Arc routing problems, Part I: The Chinese postman problem, Operations Research 43 (1995) 231–242.

[11] H.A. Eiselt, M. Gendreau, G. Laporte, Arc routing problems, Part II: The Rural postman problem, Operations Research 43 (1995) 399–414.

[12] G. Ghiani, G. Improta, G. Laporte, The capacitated arc routing problem with intermediate facilities, Networks 37 (2001) 134–143.

[13] B.L. Golden, R. Wong, Capacitated arc routing problems, Networks 11 (1981) 305–315.

[14] B.L. Golden, J.S. DeArmon, E.K. Baker, Algorithms for the capacitated Chinese postman problem, Working Paper MS/S #81-024, College of Business and Management, University of Maryland at College Park, 1981.

[15] B.L. Golden, J.S. DeArmon, E.K. Baker, Computational experiments with algorithms for a class of routing problems, Computers and Operations Research 10 (1983) 47–59.

[16] P. Greistorfer, Algorithms and implementations for the mixed capacitated Chinese postman problem, Working Paper 33, Department of Business, University of Graz, Austria, 1994.

[17] P. Lacomme, C. Prins, W. Ramdane-Chérif, Fast algorithm for general arc routing problems, in: IFORS 2002 Conference, Edinburgh, UK.

[18] P. Lacomme, C. Prins, W. Ramdane-Chérif, Competitive memetic algorithms for arc routing problems, Working

Paper 1, LOSI, Univ. Tecnologie de Troyes, France, 2001 (accepted for publication in Annals of Operations Research).
[19] J.W. Male, J.C. Liebman, Districting and routing for solid waste collection, Journal of the Environmental Engineering Division 104 (1978) 1–14.
[20] M.C. Mourão, M.T. Almeida, Lower-bounding and heuristic methods for a refuse collection vehicle routing problem, European Journal of Operational Research 121 (2000) 420–434.
[21] W.L. Pearn, Augment-insert algorithms for the capacitated arc routing problem, Computers and Operations Research 18 (1991) 189–198.
[22] G. Ulusoy, The fleet size and mix problem for capacitated arc routing, European Journal of Operational Research 22 (1985) 329–337.