PERGAMON

Computers & Operations Research 30 (2003) 705–728

computers & operations research

www.elsevier.com/locate/dsw

A cutting plane algorithm for the capacitated arc routing problem

José M. Belenguer, Enrique Benavent *

Departamento de Estadística e Investigación Operativa, Universitat de València, Dr. Moliner 50, 46100 Burjassot, Valencia, Spain

Received 1 October 2000; received in revised form 1 November 2001

Abstract

The Capacitated Arc Routing Problem (CARP) consists of finding a set of minimum cost routes that service all the positive-demand edges of a given graph, subject to capacity restrictions.

In this paper, we introduce some new valid inequalities for the CARP. We have designed and implemented a cutting plane algorithm for this problem based on these new inequalities and some other which were already known. Several identification algorithms have been developed for all these valid inequalities. This cutting plane algorithm has been applied to three sets of instances taken from the literature as well as to a new set of instances with real data, and the resulting lower bound was optimal in 47 out of the 87 instances tested. Furthermore, for all the instances tested, our algorithm outperformed all the existing lower bounding procedures for the CARP.

Scope and Purpose

In this paper, we present new developments in the Capacitated Arc Routing Problem (CARP), which has many real-world applications. Examples include the routing of refuse collection vehicles, street sweepers, snow removal vehicles and many others. Due to the complexity of the CARP, it cannot be solved optimally except for very small size instances and a number of heuristic procedures have been proposed to solve the CARP approximately. Unfortunately, it is difficult to estimate the quality of a given heuristic solution for a CARP instance unless we know a tight lower bound on its optimal cost. In this paper, we develop a cutting plane algorithm to compute a lower bound for the CARP that outperforms all the existing lower bounding procedures. This algorithm has been applied to a collection of instances taken from the literature as well as to a new set of instances with real data, and the resulting lower bounds have proven that, for 47 out of the 87 instances tested, the previously known heuristic solutions were optimal.

Keywords: Arc routing; Postman problems; Lower bounds

* Corresponding author. Tel.: +34-96-386-4354; fax: +34-96-386-4735.
E-mail address: enrique.benavent@uv.es (E. Benavent).

0305-0548/02/$ - see front matter
PII: S0305-0548(02)00046-1

1. Introduction

The Capacitated Arc Routing Problem (CARP) is a distribution problem with many practical applications. It was introduced by Golden and Wong [1] and can briefly be defined as follows: given a fleet of identical vehicles of a given capacity and an undirected graph with non-negative edge demands and edge costs, find a set of minimum cost routes for the vehicles that service all the positive-demand edges. Each route must contain the depot (a distinguished node), and the total demand it services cannot exceed the capacity of the vehicle.

The CARP is an NP-hard problem. Golden and Wong [1] showed that even the 0.5 approximate CARP (find a solution whose cost is less than 1.5 times the optimal cost) is NP-hard. Furthermore, finding a feasible solution of the CARP that uses a given number of vehicles is also NP-hard.

Several heuristic algorithms have been developed for the CARP, some of which are specially designed to deal with some of its important applications: the routing of street sweepers, refuse collection, the routing of electric meter readers, school bus routing, etc. The CARP is considered to be very difficult to solve exactly. As far as we know, the only exact method is a Branch & Bound method described in Hirabayashi et al. [2] and Kiuchi et al. [3] where, at each node, a lower bound is computed by a procedure called Node Duplication Lower Bound (NDLB). Only small instances have been solved with this method. Other lower bounding procedures for the CARP can be found in Golden and Wong [1], Assad et al. [4], Pearn [5] and Zaw Win [6]. Most of these procedures are outperformed by the ones presented in Benavent et al. [7], although, as far as we know, no Branch and Bound algorithm using these last procedures has been implemented. We refer the reader to the surveys by Assad and Golden [8] and by Eiselt et al. [9,10], and the recent book edited by Dror [11], for a detailed description of heuristics, lower bounds and applications of the CARP. Polyhedral studies of CARP and other arc routing problems are surveyed in Eglese and Letchford [12] and Benavent et al. [13].

The purpose of this paper is to present a cutting plane algorithm to compute a tight lower bound for the CARP. This algorithm is partially based on the results of the polyhedral study of the CARP presented in Belenguer and Benavent [14,15]. These studies also reported some preliminary experiences with a cutting plane algorithm that showed that the inequalities producing the largest increase in the lower bound when added to the LP relaxation are the so-called Capacity constraints and Odd Edge Cutset constraints.

In this paper we continue that line of research. We use aggregated variables to formulate the CARP and introduce new classes of valid constraints. We have implemented several procedures to identify constraints which are violated by the current LP solution and we have developed a cutting plane algorithm to compute a lower bound for the CARP. Our computational experience shows that the lower bound obtained is quite good, it outperforms all the lower bounds known for the CARP, and it can be used to asses the quality of a given heuristic solution for a CARP instance. The cutting plane algorithm has been applied to a collection of instances taken from the literature as well as to a new set of instances with real data, and the resulting lower bounds have proven that, for 47 out of the 87 instances tested, the previously known heuristic solutions are optimal.

This paper is organized in the following way. Section 2 reviews those already known inequalities that will be used in the cutting plane algorithm and presents a relaxed formulation for the CARP. In Section 3, we present three new classes of valid inequalities for the CARP.

Section 4 is devoted to the procedures used to identify violated inequalities of each class and Section 5 presents the computational results obtained on the sets of test instances mentioned above.

2. Aggregated variables for the CARP

Let $G=(V,E)$ be a connected and non-directed graph where each edge $e \in E$ has associated a cost $c_e \geqslant 0$ and a demand $d_e \geqslant 0$. The subset of edges with positive demand (called *required edges*) is denoted by R. Given a fleet of identical vehicles with capacity C, the Capacitated Arc Routing Problem (CARP) consists of finding a set of minimum cost vehicle routes that service each required edge. Each route must start and end at the *depot* (a distinguished node), and the total demand it services must not exceed the capacity C. Edges with zero demand need not be traversed by the vehicles.

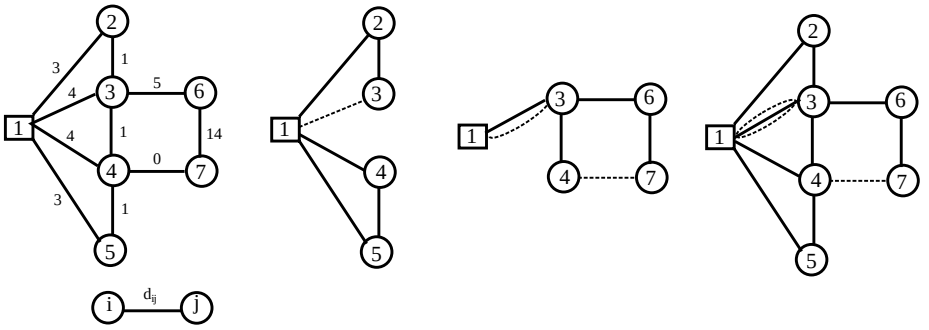
A feasible CARP solution is composed of a set of routes for the vehicles. Note that a route is a closed walk containing the depot where some edges are just traversed by that route while other edges are also serviced by it. We say that a route *deadheads* an edge whenever it traverses that edge without servicing it. We may associate with a route a family of edges containing the set of required edges that are serviced in that route and as many copies of each edge as the number of times it has been deadheaded by the route. These edge copies will be also called *deadheading edges*. We associate with every feasible CARP solution a *solution graph* containing all the required edges and all the deadheading edges contained in its set of routes.

Consider, for example the CARP instance defined by the graph G depicted in Fig. 1.a) and a vehicle capacity $C=25$; edge costs are not shown as they are not relevant to the example. A feasible CARP solution consisting of two routes is shown in Figs. 1.b) and (1.c), where solid lines indicate the edges serviced in the route and dotted lines indicate the deadheading edges. Fig. 1.d) represents the corresponding solution graph. Note that route 2 services edge (1,3) and it also deadheads once this edge.

Given a feasible CARP solution, we define, for each $e \in E$, the *aggregated variable* z_e as the number of deadheading copies of edge $e \in E$ that appear in the corresponding solution graph, i.e. z_e is the total number of times that edge e has been traversed without being serviced by the set of routes. Thus, for the feasible CARP solution depicted in Fig. 1, we have $z_{4,7}=1$, $z_{1,3}=2$, and $z_{ij}=0$ for the other edges $(i,j) \in E$.

Belenguer and Benavent [15] used a non-aggregated formulation of the CARP, i.e. the variables determine which vehicle services or just traverses each edge. They derived many classes of valid inequalities and observed that the inequalities that produce the largest increases in the lower bound when added to the LP formulation can also be written in terms of aggregated variables. This motivated us to implement a cutting plane algorithm for the CARP based on a relaxed formulation that includes only the aggregated variables z_e.

Let us introduce some new notations. For convenience, we will suppose that the *depot* is node 1. An edge $e \in E$ will be sometimes also denoted by (i,j), where i and j are its endpoints. Given a node set $S \subseteq V \backslash \{1\}$, $E(S)$ will represent the edge set of the graph induced by S, that is, $E(S) = \{(i,j) \in E: i \in S,\ j \in S\}$, and $\delta(S)$ will be the set of edges with one endpoint in S and the other not in S, that is, $\delta(S) = \{(i,j) \in E: i \in S,\ j \notin S\}$, this set also being called *edge cutset* or simply



(a) *G=(V, E)* (b) Route 1 (c) Route 2 (d) Solution graph

Fig. 1.

cut. The corresponding sets of required edges are denoted $R(S) = E(S) \cap R$, and $\delta_R(S) = \delta(S) \cap R$, respectively. Furthermore, for any subset of edges $E' \subseteq E$, we denote $z(E') = \sum_{e \in E'} z_e$.

We will say that a given route *cross a cut* $\delta(S)$, $S \subseteq V \backslash \{1\}$, whenever it traverses an edge $e \in \delta(S)$. For instance, both of the routes depicted in Figs. 1.a) and (1.b) cross the cut $\delta(S)$, for $S = V \backslash \{1\}$.

The following inequalities were introduced in Belenguer and Benavent [15] and can be expressed in terms of variables z_e.

Given a set $S \subseteq V \backslash \{1\}$, let us denote $D(S) = \sum_{e \in R(S) \cup \delta_R(S)} d_e$ and $k(S) = \lceil D(S)/C \rceil$. Note that at least $k(S)$ vehicles are required to service the demand $D(S)$, and any vehicle that services some edge in the set $R(S) \cup \delta_R(S)$ will have to cross $\delta(S)$ at least twice, so the cut $\delta(S)$ will be crossed at least $2k(S)$ times. Therefore, at least $2k(S) - |\delta_R(S)|$ deadheading copies of edges from $\delta(S)$ will exist in the solution graph associated with any feasible solution. Then, the following constraints, called *Capacity constraints*, are valid:

$$z(\delta(S)) \geqslant 2k(S) - |\delta_R(S)|, \quad S \subseteq V \backslash \{1\}. \tag{1}$$

A graph is called *even* if all its nodes have even degree. It is obvious that the solution graph associated with any feasible solution to the CARP must be an even graph; then, it can be easily shown that any edge cutset must contain an even number of edges. Therefore, for any edge cutset containing an odd number of required edges, at least one edge in the cut must be deadheaded. The so-called *Odd Edge Cutset constraints*, express this fact:

$$z(\delta(S)) \geqslant 1, \quad S \subseteq V \backslash \{1\} \quad \text{with } |\delta_R(S)| \text{ odd}. \tag{2}$$

Given $S \subseteq V \backslash \{1\}$, we will denote:

$$\alpha(S) = \begin{cases} \max\{2k(S) - |\delta_R(S)|, 1\} & \text{if } |\delta_R(S)| \text{ is odd}, \\ \max\{2k(S) - |\delta_R(S)|, 0\} & \text{if } |\delta_R(S)| \text{ is even}. \end{cases}$$

Therefore, constraints (1) and (2) can also be written in a unified form with right-hand side $\alpha(S)$.

We propose the following CARP relaxation (IP):

(IP)

$$\begin{aligned} \text{Min} \quad & \sum_{e\in E} c_e z_e & (3)\\ \text{s.t.} \quad & z(\delta(S)) \geqslant \alpha(S) \qquad \text{for all } S \subseteq V\backslash\{1\},\\ & z_e \geqslant 0 \text{ and integer} \quad \text{for all } e \in E. \end{aligned}$$

Note that the objective function (3) includes only the total deadheading cost of the solution. The real cost of a solution also involves the cost of servicing every required edge exactly once, but this is a fixed cost incurred by any feasible solution.

It will become clear in the next section that (IP) is, in general, a relaxation of the CARP because it contains integer solutions that do not correspond to any feasible CARP solution. This relaxation will be tightened by adding new valid constraints, but it is still an open question how to derive a complete formulation for the CARP that uses only the aggregated variables z_e. Notwithstanding, this relaxation is the base of a cutting plane algorithm that produces tight lower bounds at moderate computational cost.

3. New classes of valid constraints

The Capacity constraints are based on the fact that, for a subset of nodes $S \subseteq V\backslash\{1\}$, at least $k(S)$ vehicles are needed to service the demand of the edges in the set $R(S) \cup \delta_R(S)$. Note that each of those vehicles may have to service some additional demand in the path from the depot to S and in the return path, after leaving S, to the depot; but this demand has not been taken into account to compute the number of vehicles $k(S)$. Then, it may exist situations where more than $k(S)$ vehicles are needed unless some other edges, out of those in $R(S) \cup \delta_R(S)$, are deadheaded by the $k(S)$ vehicles. Let us illustrate this idea with one example.

Example 1: Recall the CARP instance depicted in Fig. 1.a) with $C=25$, and consider the following integer solution of (IP) for this instance: $z^*_{4,7} = 1$, and $z^*_{ij} = 0$, for all the other edges $(i,j) \in E$. Let $H(z^*)$ be a graph whose edge set includes all the required edges of G, and z^*_{uv} deadheading copies of each edge $(u,v) \in E$. The graph $H(z^*)$ is depicted in Fig. 2, where the deadheading edges are represented with dotted lines (the demand of each required edge is also shown). It is easy to check that this integer solution satisfies all the Capacity and Odd Edge Cutset constraints. The question that arises is whether a set of feasible routes exists that corresponds to this integer solution and, if the answer is yes, how to find them. In other words, is it possible to decompose $H(z^*)$ into two feasible routes? This question is difficult, in general, to be answered because the integer solution does not provide information about which vehicle services or just traverses each edge. Concerning our small example, the answer is negative as it can be seen by examining all the possibilities of decomposing $H(z^*)$ into two routes: in all the cases, one of the routes exceeds the capacity of the vehicle. Let us analyze why $H(z^*)$ cannot be decomposed into two feasible routes.

Let us assume, on the contrary, that a feasible CARP solution has $H(z^*)$ as the associated solution graph. Consider the node set $S = \{6,7\}$; given that $z^*(\delta(S)) + |\delta_R(S)| = 2$, the set of edges $R(S) \cup$

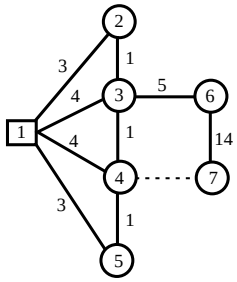


Fig. 2.

$\delta_R(S) = \{(3,6),(6,7)\}$, with a total demand of 19 units, is serviced by exactly one vehicle in this CARP solution. This vehicle has to go from the depot to node 3 or 4, service the edges (3,6) and (6,7) and return to the depot. Given that this CARP solution includes only the deadheading edge (4,7), that vehicle would have to service at least 8 additional units of demand in their way to node 3 or 4 and return. Given that $19+8 > C=25$, this route would be unfeasible, so the assumed CARP solution cannot exist.

Therefore, for any CARP solution to this instance, either the set of edges in $R(S)\cup\delta_R(S)$ is serviced by at least two vehicles or, if it is serviced by only one vehicle, it will deadhead at least one of the edges in the set $E' = \{(1,2),(1,3),(1,4),(1,5),(2,3),(3,4),(4,5)\}$. Then, for any CARP solution, either $z(\delta(S)) + |\delta_R(S)| \geqslant 4$ or $z(E') \geqslant 1$. Given that $|\delta_R(S)|$ is odd, the inequality $z(\delta(S)) \geqslant 1$ is always satisfied, so the inequality $2z(E') + z(\delta(S)) \geqslant 3$ is valid. Note that this inequality is violated by the integer solution z^*. □

The above example shows that the constraints of (IP), that is, the Capacity, Odd Edge Cutset and integrality constraints, are not sufficient to describe the feasible solutions of the CARP. It also shows how to derive a new constraint that eliminate some undesirable integer solutions. We have developed three classes of valid constraints for the CARP which are based on similar ideas.

3.1. Disjoint paths inequalities #1 (DP1)

Let $S \subseteq V\backslash\{1\}$ and let $E' \subseteq E(V\backslash S)$. We denote by T the subset of nodes of $V\backslash S$ that are endpoints of edges of $\delta(S)$. For simplicity we will suppose that $2k(S) \geqslant |\delta_R(S)|$. Note that, in this case, $\alpha(S) = 2k(S) - |\delta_R(S)|$.

We define a *Minimum Cost Flow Problem* (MCFP) on the flow graph $G' = (V\backslash S, E(V\backslash S))$ where each edge e has the following cost (b_e) and capacity (u_e):

$$b_e = \begin{cases} d_e & \text{if } e \in E', \\ 0 & \text{otherwise} \end{cases}$$

and

$$u_e = \begin{cases} 1 & \text{if } e \in E' \cap R, \\ 0 & \text{if } e \in E' \backslash R, \\ \infty & \text{otherwise.} \end{cases}$$

The depot node has a flow supply of $2k(S)$, each node of T has a flow demand equal to the number of edges of $\delta_R(S)$ incident with it, and all the other nodes have zero flow demand. If $\alpha(S) > 0$, add to G' an artificial node s, with a flow demand equal to $\alpha(S)$, and arcs from each node of T to s, with zero cost and infinite capacity. See Example 2 below.

Let $d(\text{MCFP})$ be the optimal cost of this MCFP. Roughly speaking, $d(\text{MCFP})$ is a lower bound on the demand that $k(S)$ vehicles will have to service in their way from the depot to service the edges in $R(S) \cup \delta_R(S)$, and return, assuming that no edge of E' can be deadheaded. If $d(\text{MCFP}) + D(S) > k(S)C$, then, for any feasible CARP solution, either, at least one edge of E' will be deadheaded, or more than $k(S)$ vehicles will be used to service the edges in $R(S) \cup \delta_R(S)$, so the inequality $2z(E') + z(\delta(S)) \geqslant \alpha(S) + 2$ is satisfied. This result is stated properly in the next theorem.

Theorem 3.1. *Let $S \subseteq V \backslash \{1\}$ such that $2k(S) \geqslant |\delta_R(S)|$ and let $E' \subseteq E(V \backslash S)$. If $d(\text{MCFP}) + D(S) > k(S)C$ then the following inequality*:

$$2z(E') + z(\delta(S)) \geqslant \alpha(S) + 2 \tag{4}$$

is valid for the CARP.

Proof. Given that $z(\delta(S)) + |\delta_R(S)|$ must be even and $z(\delta(S)) \geqslant \alpha(S)$, if (4) is not valid, there exists a feasible CARP solution for which $z(E') = 0$ and $z(\delta(S)) = \alpha(S) = 2k(S) - |\delta_R(S)|$. Let us assume that such solution exists, and let H be the graph associated with it. Therefore, the routes corresponding to this solution do not deadhead any edge in E' and the edges in $R(S) \cup \delta_R(S)$ are serviced by exactly $k(S)$ vehicles.

Let us consider the paths followed by these $k(S)$ vehicles from the depot to their first visit to a node in S, and from their last visit to a node in S up to the depot. We may associate with each one of these paths a path in H: with every edge serviced in that path we associate the corresponding required edge in H, and with every edge deadheaded in that path, we associate a deadheading copy of that edge in H. We can identify in this way $2k(S)$ edge disjoint paths in H, each one joining the depot to a node of S.

Let $\delta^H(S)$ be the set of edges in H with one endpoint in S and the other in $V \backslash S$. Each one of these paths contains exactly one edge of $\delta^H(S)$. Let $P_1, P_2, \ldots, P_{2k(S)}$ be the paths that result from the above $2k(S)$ paths after removing from them the edges of $\delta^H(S)$. For every node $u \in T$, let α_u be the number of deadheading edges of $\delta^H(S)$ incident with u. Obviously, the number of paths ending at a node $u \in T$ is equal to $|\delta(\{u\}) \cap \delta_R(S)| + \alpha_u$, and $\sum_{u \in T} \alpha_u = \alpha(S)$. Furthermore, given that $z(E') = 0$, the paths $P_1, P_2, \ldots, P_{2k(S)}$, cannot contain any deadheading copy of an edge of E'.

Let us consider these paths as directed paths from the depot to a node of T. We define from this set of paths the following flow in G': for each $(i,j) \in E(V \backslash S)$, let y_{ij} (y_{ji}) be the number of times

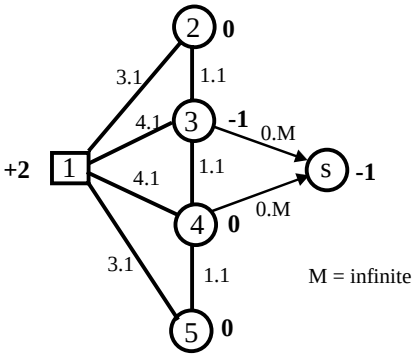


Fig. 3.

edge (i,j) is traversed from i to j (from j to i) in the set of paths, either as a serviced edge or as a deadheading edge, and let $y_{us} = \alpha_u$ for each $u \in T$. It can be easily checked that this is a feasible flow for the above defined MCFP. The cost of this flow is equal to the total demand serviced in E' by the paths $P_1, P_2, \ldots, P_{2k(S)}$, and is greater than or equal to the minimum cost flow $d(\text{MCFP})$. Given that $d(\text{MCFP}) + D(S) > k(S)C$, we get a contradiction: the $k(S)$ routes that service the edges in $R(S) \cup \delta_R(S)$ would service a total demand that exceed their total capacity.

Therefore we conclude that, for any feasible CARP solution, either it will deadhead at least one edge in E' (that is $z(E') \geqslant 1$), or it will use more than $k(S)$ vehicles to service the demand in $R(S) \cup \delta_R(S)$, so (4) is valid. □

Example 2: Consider the CARP instance of Example 1 (Fig. 1.a)). Fig. 3 shows the MCFP that corresponds to the set of nodes $S = \{6,7\}$ and the set of edges $E' = \{(1,2),(1,3),(1,4),(1,5),(2,3),(3,4),(4,5)\}$. Cost and capacity are shown besides each edge, and the number besides each node indicates its flow supply/demand (flow demands are indicated with negative numbers). Note that $d(\text{MCFP}) = 8$, and $8 + 19 > C = 25$, so the inequality $2z(E') + z(\delta(S)) \geqslant 3$ is valid. □

Improvement of inequalities DP1: Note that the validity of inequalities DP1 depends on the choice of an appropriate set of edges E'. Nevertheless, the smaller that E' is, the stronger is the inequality we obtain. To check if a given edge e can be removed from E' without affecting the validity of the inequality, we may solve a new MCFP that corresponds to the set $E' \backslash \{e\}$. If the inequality (4) is still valid, we remove e from E'. This can be done sequentially with every edge in E'. For instance, note that edges (2,3) and (3,4), that are included in the set E' of Example 2, can be removed from E' because the value of the new MCFP would be 7 and $7 + 19 > 25$.

On the other hand, the coefficients of the edges that remain in E' can be decreased by parity considerations, also obtaining a stronger inequality. Note that the coefficients of variables z_e, $e \in E'$, in (4) are equal to 2 because it may exists a feasible CARP solution for which $z(E') = 1$ and $z(\delta(S)) = \alpha(S)$. Note that if $\delta(S')$, $S' \subseteq V \backslash \{1\}$, is an even edge cutset (that is, $|\delta_R(S')|$ is even), then, by the parity requirement, the total number of times that any feasible solution deadheads edges of $\delta(S')$ is even. Therefore, in this case, $z(\delta(S')) > 0$ implies that $z(\delta(S')) \geqslant 2$. Thus, if $\delta(S')$ is an even edge cutset contained in E', then for any $e \in \delta(S')$, $z_e = 1$ implies $z(E') \geqslant 2$, so the coefficient of z_e in the inequality (4) can be decreased from 2 to 1, without affecting its validity. For instance,

it can be checked that every edge in $E' = \{(1,2),(1,3),(1,4),(1,5),(4,5)\}$ belongs at least to one even edge cutset which is contained in E', then their coefficients can be equal to 1 in the inequality. Therefore, an improved DP1 inequality in Example 2 is $z_{1,2}+z_{1,3}+z_{1,4}+z_{1,5}+z_{4,5}+z_{3,6}+z_{4,7} \geqslant 3$.

3.2. Disjoint paths inequalities #2 (DP2)

Let $S \subseteq V\backslash\{1\}$ and let r be an integer number, $1 \leqslant r \leqslant k(S)$. We define:

$$\beta(S,r) = \begin{cases} \text{sum of the demands of the } 2r-\alpha(S) \text{ least demand edges} & \\ \text{in the set } \delta_R(S) & \text{if } 2r-\alpha(S) > 0, \\ 0 & \text{if } 2r-\alpha(S) \leqslant 0. \end{cases}$$

Consider a feasible solution to the CARP such that the number of times that it deadheads edges in $\delta(S)$ is exactly $\alpha(S)$ (that is, it satisfies $z(\delta(S)) = \alpha(S)$). Then, if r routes in that solution cross the cut $\delta(S)$, they will service at least $\beta(S,r)$ units of demand from the edge set $\delta_R(S)$. Consider for instance the graph of Fig. 1.a) and the set $S = \{2,3,4,5,6,7\}$, for which $\alpha(S) = 0$, then $\beta(S,1) = 6$. This means that, if a CARP solution satisfies $z(\delta(S)) = 0$, one vehicle ($r = 1$) whose route cross the cut $\delta(S)$ will service at least 6 units of demand in $\delta_R(S)$.

Theorem 3.2. *Consider a sequence of subsets* $S_0 \subset S_1 \subset \cdots \subset S_t \subseteq V\backslash\{1\}$, *with* $t \geqslant 1$, *such that*:
(i) $2k(S_0) \geqslant |\delta_R(S_0)|$,
(ii) $\delta(S_i) \cap \delta(S_{i+1}) = \emptyset$, *for* $i = 0,\ldots,t-1$,
(iii) $\beta(S_i, k(S_0)) > 0$ *for* $i = 1,\ldots,t$, *and*
(iv) $D(S_0) + \sum_{i=1}^{t} \beta(S_i, k(S_0)) > k(S_0)C$.
Then, the following inequality is valid for the CARP:

$$\sum_{i=0}^{t} z(\delta(S_i)) \geqslant \sum_{i=0}^{t} \alpha(S_i) + 2. \tag{5}$$

Proof. Given that the solution graph must be even and the inequalities $z(\delta(S_i)) \geqslant \alpha(S_i)$ for $i=0,\ldots,t$, are all valid, if $z(\delta(S_i)) > \alpha(S_i)$ holds just for one $i \in \{0,\ldots,t\}$, then $z(\delta(S_i)) \geqslant \alpha(S_i) + 2$, so the inequality (5) would be satisfied. Then, if we assume that (5) is not valid, it will exist a feasible CARP solution which satisfies $z(\delta(S_i)) = \alpha(S_i)$ for $i = 0,\ldots,t$. In particular, $z(\delta(S_0)) = \alpha(S_0) = 2k(S_0) - |\delta_R(S_0)|$ (recall condition (i)), therefore, only $k(S_0)$ vehicles cross the cut $\delta(S_0)$ to service the edges in the set $R(S_0) \cup \delta_R(S_0)$. Note that these $k(S_0)$ vehicles will also have to cross all the cuts $\delta(S_i)$ for $i = 1,\ldots,t$, to reach S_0 from the depot. Then, $\sum_{i=1}^{t} \beta(S_i, k(S_0))$ is a lower bound on the demand serviced by them while crossing these cuts. Nevertheless, condition (iv) implies that the demand $D(S_0) + \sum_{i=1}^{t} \beta(S_i, k(S_0))$ cannot be serviced by $k(S_0)$ vehicles, so the assumed feasible CARP solution cannot exist. Therefore, for every feasible solution, $z(\delta(S_i)) \geqslant \alpha(S_i) + 2$ will hold for at least for one $i \in \{0,\ldots,t\}$, so (5) is valid. □

Example 3: Consider the CARP instance of Fig. 4.a), with $C = 20$. The integer solution: $z^*_{1,2} = 1$, $z^*_{3,4} = 1$, and $z^*_{ij} = 0$, for the other edges $(i,j) \in E$, satisfies all the Capacity and Odd Edge

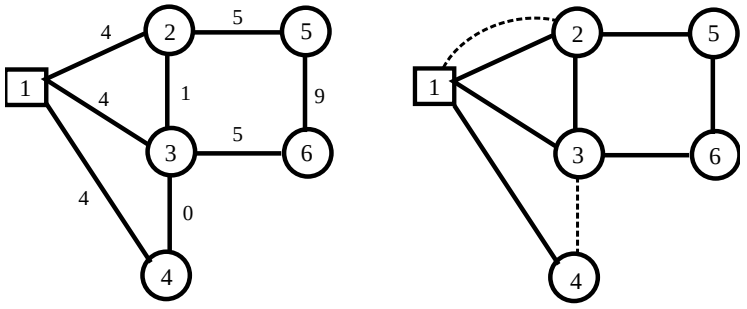


Fig. 4.

Cutset constraints as it can be easily verified. The graph $H(z^*)$ as defined in Example 1 is depicted in Fig. 4.b). Consider the following node sets:

$S_0 = \{5, 6\}$ with $D(S_0) = 19$, $k(S_0) = 1$ and $\alpha(S_0) = 0$, and

$S_1 = \{2, 3, 4, 5, 6\}$ with $\alpha(S_1) = 1$ and $\beta(S_1, 1) = 4$.

Then $D(S_0) + \beta(S_1, k(S_0)) = 19 + 4 = 23 > k(S_0)C = 20$, so the inequality

$z(\delta(S_0)) + z(\delta(S_1)) \geqslant 3$

is valid. Note that the above solution z^* violates this constraint. □

It is interesting to note that no DP1 inequality can be found that cut the integer solution defined in Example 3. On the other hand, it easy to show that no DP2 inequality can eliminate the integer solution of Example 1. Therefore, although they are based on the same idea, both classes of inequalities are needed to eliminate integer solutions that do not correspond to feasible CARP solutions.

3.3. Disjoint paths inequalities #3 (DP3)

These inequalities have been derived by combining the ideas that lead to inequalities DP1 and DP2. In fact, inequalities DP3 generalize both DP1 and DP2, but we have decided to present them separately to make the explanation clearer and also because the heuristics designed to identify each class of constraints are different.

Consider a sequence of node subsets $S_0 \subset S_1 \subset \cdots \subset S_t \subseteq V\backslash\{1\}$, with $t \geqslant 1$, and an edge subset $E' \subseteq E(V\backslash S_0)$, satisfying the following conditions:

$$2k(S_0) \geqslant |\delta_R(S_0)|, \tag{6}$$

$$\delta(S_i) \cap \delta(S_{i+1}) = \emptyset \quad \text{for } i = 0, \ldots, t-1, \tag{7}$$

$$E' \cap \delta(S_i) = \emptyset \quad \text{for } i = 1, \ldots, t \tag{8}$$

Let T_i (respectively U_i), for $i=0,\ldots,t$, be the subset of nodes of $V\backslash S_i$ (respectively S_i) incident with edges of $\delta(S_i)$. Consider the Minimum Cost Flow Problem (MCFP) defined on a flow graph $G'=(N,A)$ constructed as follows:

(a) If $\alpha(S_i)=0$ for all $i=1,\ldots,t$,
  (a.1) $N=(V\backslash S_0)$ and $A=E(V\backslash S_0)$.
  (a.2) Each edge $e\in A$ has the following cost (b_e) and capacity (u_e):

$$b_e=\begin{cases} d_e & \text{if } e\in E'\cup\left(\bigcup_{i=1}^{t}\delta_R(S_i)\right),\\ 0 & \text{otherwise}\end{cases}$$

and

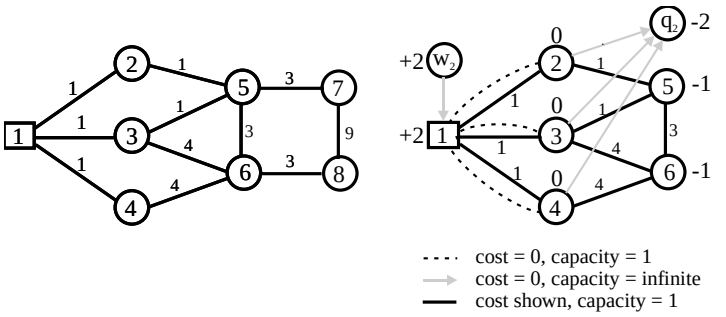
$$u_e=\begin{cases} 1 & \text{if } e\in\left(\left(E'\cup\left(\bigcup_{i=1}^{t}\delta(S_i)\right)\right)\cap R\right),\\ 0 & \text{if } e\in\left(\left(E'\cup\left(\bigcup_{i=1}^{t}\delta(S_i)\right)\right)\backslash R\right),\\ \infty & \text{otherwise.}\end{cases}$$

  (a.3) The depot (node 1) has a flow supply equal to $2k(S_0)$, each node in T_0 has a flow demand equal to the number of edges of $\delta_R(S_0)$ incident with it, and all the other nodes have zero flow demand.
  (a.4) If $\alpha(S_0)>0$, add an artificial node s with flow demand $\alpha(S_0)$, and arcs from each node of T_0 to s, with zero cost and infinite capacity.
(b) Otherwise, construct the graph G' as in (a) and, for every $i=1,\ldots,t$, such that $\alpha(S_i)>0$, add to G' the following nodes, edges and arcs:
  (b.1) Add an artificial copy of every edge $e\in\delta(S_i)$ with zero cost and capacity $\alpha(S_i)$.
  (b.2) If $|\delta(S_i)|>1$, add one artificial source, w_i, with supply $(|\delta(S_i)|-1)\alpha(S_i)$, and one artificial sink, q_i, with flow demand $(|\delta(S_i)|-1)\alpha(S_i)$. Add also arcs, with zero cost and infinite capacity, from w_i to every node of T_i and from every node of U_i to q_i. (See Example 4 and Fig. 5 below).

Theorem 3.3. *Let $S_0\subset S_1\subset\cdots\subset S_t\subseteq V\backslash\{1\}$, $t\geqslant 1$, be a sequence of node subsets, and let $E'\subseteq E(V\backslash S_0)$ be an edge subset such that* (6), (7) *and* (8) *are satisfied. Let d*(MCFP) *be the optimum value of the MCFP defined above. If d*(MCFP) $+D(S_0)>k(S_0)C$, *then the following inequality is valid for the CARP*:

$$\sum_{i=0}^{t} z(\delta(S_i))+2z(E')\geqslant\sum_{i=0}^{t}\alpha(S_i)+2. \tag{9}$$

Proof. By contradiction. Suppose that a feasible CARP solution exists that deadheads no edge in E' and satisfies all the valid constraints $z(\delta(S_i))\geqslant\alpha(S_i)$, for $i=0,\ldots,t$, with equality. Let H be the



(a) *G=(V,E)* (b) Minimum Cost Flow Problem

Fig. 5.

graph associated with this solution, and let $\delta^H(S)$ be the set of edges in H with one endpoint in S and the other in $V\backslash S$, where $S \subseteq V\backslash\{1\}$. Note that these assumptions imply that graph H contains $\alpha(S_i)$ deadheading edges in each edge cutset $\delta(S_i)$, for $i = 0,\ldots,t$ and only $k(S_0)$ vehicles has been used to service the edges in the set $R(S_0) \cup \delta_R(S_0)$.

Following the lines of the proof of Theorem 3.1, we can identify in H $2k(S_0)$ edge disjoint paths joining the depot to a node in the set T_0. These paths can contain at most $\alpha(S_i)$ deadheading edges in each $\delta^H(S_i)$, for $i = 1,\ldots,t$. From these paths we may build a feasible flow in G' in the same way than in the proof of Theorem 3.1, except for the flow through the artificial edges and arcs added in (b.1) and (b.2), and their corresponding original edges, which is constructed as follows. Assign to every edge of G' that is an artificial copy of an edge $(u,v) \in \delta(S_i)$, a flow, in the appropriate direction, equal to the number of deadheading copies of (u,v) that has been used by the $2k(S_0)$ paths, and assign a unit flow to the corresponding original edge whenever it is serviced in this set of paths. This flow satisfies the flow supplies and demands of graph G' except for the artificial nodes w_i and q_i, $i = 1,\ldots,t$. Note that the total flow through the artificial edges coming from edges in $\delta(S_i)$ is at most $\alpha(S_i)$, so their total residual capacity is at least $(|\delta(S_i)| - 1)\alpha(S_i)$. Then, we may satisfy the flow supply and demand of the artificial nodes w_i and q_i, by sending $(|\delta(S_i)| - 1)\alpha(S_i)$ additional units of flow from w_i to q_i, for $i = 1,\ldots,t$, which completes the construction of the feasible flow.

The total cost of this flow is equal to the total demand serviced in E' by the $2k(S_0)$ paths and it is greater than or equal to the minimum cost flow, $d(\text{MCFP})$. Therefore, if $d(\text{MCFP}) + D(S_0) > k(S_0)C$ holds, the routes of the $k(S_0)$ vehicles servicing $R(S_0) \cup \delta_R(S_0)$ would not be feasible. Then, at least one of the above assumptions is false, so the inequality (9) is valid. □

Example 4: Consider the CARP instance of Fig. 5.a) with vehicle capacity $C = 20$, where the demands are depicted besides each edge (edge costs are not shown as they are not relevant to the example), and the following sets:

$$S_0 = \{7,8\}, \quad D(S_0) = 15, \quad k(S_0) = 1, \quad \alpha(S_0) = 0,$$

$$S_1 = \{5,6,7,8\}, \quad \alpha(S_1) = 0,$$

$$S_2 = \{2,3,4,5,6,7,8\}, \quad \alpha(S_2) = 1,$$

$$E' = \{(5,)\}.$$

Fig. (5.b) represents the flow graph for the MCFP as defined above. The number besides each node represents its supply (flow demand has been represented as negative supply).

Given that $\alpha(S_2) = 1 > 0$, one artificial source and sink (nodes w_2 and q_2) as well as artificial copies of the edges of $\delta(S_2)$ (represented with dotted lines) have been added. It is easy to see that an optimal solution of the MCFP is as follows:

$$f(w_2,1) = 2,$$

$$f^*(1,2) = f(1,2) = f^*(1,3) = f^*(1,4) = f(2,q_2) = f(4,q_2) = f(2,5) = f(3,6) = 1,$$

and $f(i,j) = 0$ for the other edges, where $f(i,j)$ represents the flow from i to j and $f^*(i,j)$ represents the flow from i to j through the artificial copy of edge (i,j). The flow cost is 6, so $D(S_0) + d(\text{MCFP}) = 15 + 6 > 20$. Thus, the following constraint is valid for this CARP instance:

$$z(\delta(S_0)) + z(\delta(S_1)) + z(\delta(S_2))2z_{5,6} \geqslant 1 + 2 = 3.$$

It can be easily shown that the solution given by: $z^*_{1,3} = 1$, and $z^*_{ij} = 0$, for all $(i,j) \in E \backslash \{(1,3)\}$, satisfies all the Capacity and Odd Edge Cutset constraints while it violates this last inequality. It is also interesting to note that no violated inequality of types DP1 or DP2 can be found for this solution. □

Improvement of inequality DP3. Inequalities DP3 can be improved in a similar way as it was explained for inequalities DP1: some edges can be removed from E' and the coefficients of z_e, for some $e \in E'$, in the inequality (9) can be decreased from 2 to 1. Furthermore, the condition for decreasing these coefficients can be generalized.

Let $E^A = E' \cup (\bigcup_{i:\alpha(S_i)=0} \delta(S_i))$, and let $\delta(S'), S' \subseteq V \backslash \{1\}$, be an even edge cutset contained in E^A. Then, if $e \in \delta(S') \cap E'$, $z_e = 1$ implies $z(\delta(S')) \geqslant 2$, because $\delta(S')$ is an even cutset, and therefore $z(E^A) \geqslant 2$. Given that E^A contains the set E' and only those edge cutsets S_i for which $\alpha(S_i) = 0$, this implies that the inequality that results from (9) by decreasing the coefficient of z_e from 2 to 1, will be satisfied. Therefore, the coefficients of all the edges in $E' \cap \delta(S')$ in the inequality (9) can be decreased from 2 to 1 and the inequality is still valid.

4. A cutting plane algorithm for the CARP

The cutting plane algorithm can briefly be described as follows. Initially, we build a linear program (LP) containing the non-negativity constraints and a subset of inequalities (1) and (2) (details are given below). Then, at each iteration, we solve the current LP, we look for a set of valid inequalities of types (1), (2), (4), (5) and (9), violated by the optimal LP solution, we add them to the LP and proceed as before.

The optimal cost of the LP at any iteration is a lower bound for the cost of the optimal CARP solution. The algorithm stops when either no violated inequality is found, or the lower bound is equal to a known upper bound provided by a heuristic algorithm for the CARP. In the latter case, the

lower bound and the corresponding heuristic solution are optimal, otherwise it would be necessary to employ a branching scheme to solve exactly the CARP instance.

Note that the cutting plane algorithm alone cannot solve optimally any CARP instance because, even if it produces an integer solution, say z^*, it does not contain information about which vehicle services or traverses each edge, so we cannot know if there exists a feasible CARP solution corresponding to z^*. Only if we know a heuristic solution, provided by any heuristic algorithm for the CARP, with the same cost than z^*, we may conclude that the corresponding lower bound is optimal. From a practical point of view, the cutting plane algorithm can be used to estimate the quality of a given heuristic CARP solution. As the computational results will show, that estimation is rather good because the lower bound obtained with the cutting plane algorithm is very tight on average.

The details of the cutting plane algorithm are presented in what follows. The initial LP includes: the non-negativity constraints, the Odd Edge Cutset constraints (2) for the sets $S = \{v\}$, where v is an *odd node* (incident with an odd number of required edges) and constraints (1) and (2) for a family of node sets that are generated by the following procedure:

Sequence of edge cutsets procedure: 1. Set $W = \{1\}$.

2. Let $S = V \backslash W$, compute $\alpha(S)$ and, if $\alpha(S) > 0$, generate inequality (1) or (2).

3. Add to W those nodes adjacent to at least one node of W. If $W = V$, stop; otherwise go to 2.

This procedure can be viewed as a cheap way of generating some initial Capacity/Odd Edge Cutset constraints. In the case of the graph induced by the required edges not being connected, we also generate constraints (1) for all node sets of the connected components of that graph.

At each iteration of the cutting plane algorithm we have to use several identification algorithms to generate valid inequalities that are violated by the current LP solution. The efficiency of the cutting plane algorithm relies, to a great extent, on the design of effective procedures for this purpose. The *Separation* (or *Identification*) *Problem* associated with any class of valid inequalities is defined as the problem of finding an inequality of that class which is violated by the LP solution or proving that none exists. Concerning the inequalities considered in this paper, the Odd Edge Cutset constraints are the only class of inequalities for which a polynomial algorithm is known that solves the Identification problem. For the other classes, we have designed heuristic algorithms that have proved to be very effective for finding violated inequalities.

In the following subsections, we present the identification procedures for each class of valid constraints for the CARP that we have used in the cutting plane algorithm. The identification procedures for Capacity and Odd Edge Cutset constraints are used at every iteration and, only in those iterations where no violated inequality of these classes is found, the identification procedures for Disjoint Path inequalities DP1, DP2 and DP3 are applied. All the violated inequalities found at a given iteration are added to the LP.

4.1. Identification of capacity and odd edge cutset constraints

Let $\bar{z}$ be the optimal LP solution at the current iteration of the cutting plane algorithm and let $G(\bar{z})$ be the graph induced in G by the edges $e \in E$ such that $\bar{z}_e > 0$.

The Identification problem for the Odd Edge Cutset constraints (2) can be solved in polynomial time using the Odd Minimum Cut algorithm of Padberg and Rao [16]. The procedure is as follows: consider the graph $G(\bar{z})$ and give a capacity $\bar{z}_e$ to each edge $e \in E$; moreover, label as odd nodes

those that are odd nodes of G (incident with an odd number of required edges). An *odd cut* is defined as an edge cutset $\delta(S)$ such that S contains an odd number of odd labeled nodes. Using the algorithm of Padberg and Rao we find the minimum Capacity Odd Cut in $G(\bar{z})$. If the capacity of this cut is less than 1, the corresponding Odd Edge Cutset constraint is violated, otherwise the capacity of any odd cut is at least 1 so no violated constraint of this type exists.

The identification problem for the Capacity constraints (1) seems to be more difficult. We have developed several heuristic algorithms to identify this class of constraints; all of them generate node sets $S \subseteq V\backslash\{1\}$ for which (1) and (2) are checked for possible violation.

One simple heuristic consists of computing the connected components of $G(\bar{z})$ and checking (1) and (2) for the corresponding sets of nodes. The following heuristics are more elaborate.

Fractional capacity identification procedure: The following inequalities are called Fractional Capacity constraints:

$$z(\delta(S)) \geqslant 2D(S)/C - |\delta_R(S)|, \quad S \subseteq V\backslash\{1\}, \tag{10}$$

They are clearly dominated by the Capacity constraints (1) because $k(S) = \lceil D(S)/C \rceil$. The interest in (10) relies on the fact that they can be identified in polynomial time. These constraints (as well as the method to identify them) are similar to the ones developed by Harche and Rinaldi [17] for the CVRP. The procedure consists of solving a maximum flow problem on a new graph $G'(\bar{z})$ constructed from G by adding an artificial node, denoted $n+1$, connected to every node i of G, $i = 1, \ldots, n$, where n denotes the number of nodes of G. The capacity of each edge $e = (i, j)$ in $G'(\bar{z})$ is denoted by b_{ij} and is defined as follows:

$$b_{ij} = \begin{cases} \bar{z}_e & \text{if } e \in E\backslash R, \\ \bar{z}_e + 1 - \dfrac{d_e}{C} & \text{if } e \in R, \\ \dfrac{1}{C} \displaystyle\sum_{f \in \delta_R(\{i\})} d_f & \text{if } i \in \{1, \ldots, n\} \text{ and } j = n+1. \end{cases}$$

Let $S \subseteq V\backslash\{1\}$ be a set of original nodes. If we subtract $P = (2/C)\sum_{e \in R} d_e$ to the capacity of the cut defined by $S \cup \{n+1\}$ in $G'(\bar{z})$, we have:

$$\begin{aligned} &\sum_{(i,j) \in \delta_R(S)} b_{ij} + \sum_{(i,j) \in \delta(S)\backslash\delta_R(S)} b_{ij} + \sum_{i \in V\backslash S} b_{i,n+1} - \frac{2}{C}\sum_{e \in R} d_e \\ &\quad = |\delta_R(S)| + \bar{z}(\delta(S)) - \frac{1}{C}\sum_{e \in \delta_R(S)} d_e + \frac{1}{C}\sum_{i \in V\backslash S}\sum_{f \in \delta_R(\{i\})} d_f - \frac{2}{C}\sum_{e \in R} d_e \\ &\quad = |\delta_R(S)| + \bar{z}(\delta(S)) + \frac{2}{C}\sum_{e \in R(V\backslash S)} d_e - \frac{2}{C}\sum_{e \in R} d_e = |\delta_R(S)| + \bar{z}(\delta(S)) - \frac{2D(S)}{C}. \end{aligned}$$

That is, if we subtract P to the capacity of the cut defined by $S \cup \{n+1\}$ we obtain the slack of constraint (10) for node set S. As it is well known, the minimum capacity cut that separates nodes 1 and $n+1$ can be obtained by solving a maximum flow problem. Let v^* be the capacity of that

minimum capacity cut, and let S^* be the set of original nodes defining this cut in the shore of $n+1$. If $v^* - P < 0$, constraint (10) is violated (and also (1)) for S^*; otherwise, no constraint of type (10) is violated.

Given that constraint (1) is stronger that constraint (10), it may happen that constraint (1) is violated for set S^* even if (10) is not. Then, this procedure is used to generate a set S^* for which (1) and (2) are checked for possible violation.

Perturbed demand procedure: For any set $S \subseteq V\backslash\{1\}$, the slack of the constraint (10) is always greater than that of the Capacity constraint (1). One heuristic method that could approximate the slack of (10) to that of (1) is to compute the slack of (10) in a graph where the demands have been increased by a given percentage. The Perturbed Demand procedure consists of substituting the demand of each edge e by $d'_e = (1 + p)d_e$, where $0 < p < 1$, and applying the Fractional Capacity identification procedure with these perturbed demands. This procedure is applied using ten different values of p.

4.2. Identification of disjoint path inequalities

The identification of violated Disjoint Path inequalities is far more complex than that of the previous ones. The strategy we have followed to generate inequalities of these classes is to use the information obtained while trying to identify Capacity and Odd Edge Cutset constraints. Thus, the identification procedures described in Section 4.1 generate a number of tentative edge cutsets that are stored (in fact we store their corresponding node sets $S \subseteq V\backslash\{1\}$) in a pool to be used in the identification of Disjoint Path inequalities. From this pool, three lists of node sets are built: INI, a list of node sets that could be used to define an inequality of type DP1; and SEQ2 and SEQ3, two lists that contain sequences of node sets that are used as possible configurations for inequalities DP2 and DP3, respectively.

Let D_T denote the total demand of the required edges and let us denote $K^* = \lceil D_T/C \rceil$, which is a lower bound on the minimum number of vehicles used by any feasible solution to the CARP.

Each time any identification procedure of those described in Section 4.1 generates a node set $S \subseteq V\backslash\{1\}$ which is different from those previously generated, it is stored in the pool and lists INI, SEQ2 and SEQ3 are updated by performing the following steps:

Step 1: An attempt is made to insert the node set S in every sequence of node sets contained in the list SEQ3. Let $S_0, S_1, \ldots, S_t$ be one sequence of SEQ3 satisfying $k(S) \geqslant k(S_0)$ and $\beta(S, k(S_0)) > 0$:

- If there exists $j \in \{0, \ldots, t-1\}$ such that $S_j \subseteq S \subseteq S_{j+1}$ and $\delta(S) \cap \delta(S_{j+1}) = \delta(S_j) \cap \delta(S) = \emptyset$, a new sequence is generated by inserting set S between S_j and S_{j+1}.
- If $S_t \subseteq S$ and $\delta(S_t) \cap \delta(S) = \emptyset$, a new sequence is generated by appending S at the end of the sequence.

For any new sequence generated, if it satisfies condition (iv) of the statement of theorem 3.2, it defines a configuration for a valid inequality of type DP2, so it is added to list SEQ2; otherwise, it is added to list SEQ3.

Step 2: If $k(S) < K^*$ and $2k(S) \geqslant |\delta_R(S)|$, node set S is added to INI and we generate all possible sequences of node sets having S as the first set of the sequence, that is $S_0 = S$, and the other sets in

the pool as possible intermediate sets. A similar procedure to that of Step 1 is used for this purpose. The sequences generated are added either to list SEQ2 or to list SEQ3 as in Step 1.

The following procedures use the lists INI, SEQ2 and SEQ3 to look for violated Disjoint Path inequalities. The procedures are called on in the same order as presented below, and once one violated inequality is found, the identification process is stopped.

Identification procedure for DP2: Any sequence of node sets in the list SEQ2 defines a valid inequality of type DP2. If it is violated by the LP solution, it is added to the LP and removed from SEQ2.

Identification procedure for DP1 and DP3: Inequalities of types DP1 and DP3 are generated and checked at the same time using lists INI and SEQ3. First, a node set, say S, from the list INI is selected and an attempt is made to generate a valid inequality DP1 using this set; thereafter, we attempt to generate a valid inequality of type DP3 from every sequence of node sets in list SEQ3 having S as the initial set.

Let S be a node set from list INI. We consider only those node sets for which $\bar{z}(\delta(S)) < \alpha(S)+2$ holds, because otherwise any valid inequality DP1 or DP3 generated from S would be satisfied by the LP solution.

We first try to build from S an inequality DP1. We define $E' = \{e \in E(V\backslash S)\colon \bar{z}_e = 0\}$ and solve a minimum cost flow problem (MCFP) as described in Section 3.1. Let $d(\mathrm{MCFP})$ be the cost of the optimal flow: if $d(\mathrm{MCFP}) + D(S) > k(S)C$ then the DP1 inequality (4) is valid and violated by the LP solution, and the process is completed. Otherwise, we consider every sequence of node sets in SEQ3 whose initial set is S and try to build an inequality of type DP3 as follows.

Let $S = S_0, S_1, \ldots, S_t$ be one sequence in SEQ3. We first check that the corresponding inequality could be violated, that is, $\sum_{i=0}^{t} \bar{z}(\delta(S_i)) < \sum_{i=0}^{t} \alpha(S_i) + 2$; then, we define the set $E' = \{e \in E(V \setminus S_0)\backslash \bigcup_{i=1}^{t} \delta(S_i)\colon \bar{z}_e = 0\}$ and solve a MCFP as described in Section 3.3. Let $d(\mathrm{MCFP})$ be the cost of the optimal flow: if $d(\mathrm{MCFP}) + D(S_0) > k(S_0)C$ holds, inequality (9) is valid and violated.

4.3. Improvement procedure for inequalities DP1 and DP3

Whenever a violated inequality of type DP1 or DP3 is found, we try to improve it by reducing from 2 to 1 the coefficients of some edges in the set E' associated with these inequalities, or just removing them, as it was explained at the end of Sections 3.1 and 3.3.

To check if a given edge e can be removed from the edge set E' without affecting the validity of the inequality, we solve a new MCFP that corresponds to the set $E'\backslash\{e\}$. If the inequality is still valid, we remove e from E'. This procedure is applied sequentially to each edge in E'. It has been observed that the set of edges that finally remain in E' depend on the order in which they are processed, so different inequalities could be obtained by trying with different orders. Nevertheless, we apply the method only once, using an arbitrary order.

Given that inequalities DP1 are included in DP3, we describe the coefficient reduction procedure only for inequalities DP3. Recall that these inequalities are determined by a sequence of node sets $S_0, S_1, \ldots, S_t$, and an edge set E'. Let us denote $E^A = E' \cup (\bigcup_{i:\alpha(S_i)=0} \delta(S_i))$, then, the coefficient of z_e, $e \in E'$, in (9), can be reduced from 2 to 1 whenever it can be found an even edge cutset $\delta(S') \subseteq E^A$ such that $e \in \delta(S')$. We use the following method to generate appropriate edge cutsets. First, we identify the node sets of the connected components of the graph $G\backslash E^A$. Then, these node sets are

grouped arbitrarily in different ways, thus producing several edge cutsets that are contained in E^A and those which are even are used in the coefficient reduction.

Note that the coefficient reduction method depends on the actual set E'; therefore, we first try to remove as many edges of E' as possible and then we apply the coefficient reduction method with the resulting set E'.

5. Computational results

Except for the code of the Minimum Cost Flow Problem, which was written in FORTRAN 77 by Bertsekas and Tseng [18], all the other algorithms were coded in C and run on a SUN Sparc 20. We have used the lp-solver CPLEX 3.0 [19].

Four sets of instances were used to test our method. The first set of instances comes from Benavent et al. [7] and contains 34 instances which are defined on 10 different graphs; for each graph I, several instances were generated (denoted I.A, I.B, ...) by changing the capacity of the vehicles. The second set contains 23 instances generated by Golden et al. [20] that are denoted gdb1, ..., gdb23. The third set consists of 6 instances (denoted kshs1 to kshs6) that were used by Kiuchi et al. [3] to test a parallel Branch & Bound method. The instances in these three sets were randomly generated (usually by hand) and all their edges are required edges.

A fourth set of instances, referred to as Eglese instances, has been constructed from real data. The real data represents the road network of two areas (east and south) of the county of Lancashire (UK) which were digitized for a study of winter gritting (see Li [21] and Li and Eglese [22]). Actually, the study also included the north area, but we have not used this because its size was excessive. The data sets included for each road segment their length and a category number from 1 to 4. Category 1 roads needed to be treated within two hours, category 2 within four hours, category 3 would be treated later if it were still necessary and category 4 roads are those which do not need treatment at all. From each data set (edge set), we have considered four possible subsets of required edges; the ith set, $i = 1, \ldots, 4$, includes as required edges all the edges in categories $j = 1, \ldots, i$. Both the cost and the demand of each edge were set equal to their length (in hectometers). Finally, three CARP instances were generated for each case, by considering different vehicle capacities. Thus, a total number of 24 instances have been generated. For example, each of the instances denoted by egl-e2-A, egl-e2-B, and egl-e2-C, contain all the edges from the east area, and those of categories 1 and 2 are required edges, but they differ in their vehicle capacities. The Eglese instances have sizes considerably greater than those in the preceding sets: up to 140 nodes and 190 edges, and a number of vehicles up to 35; furthermore, many of these instances include non-required edges.

The complete data of these instances can be obtained on request from the authors or directly from http://www.uv.es/~belengue/carp.html.

Tables 1–4 present the results of the cutting plane algorithm when applied to these four sets of instances. The first four columns of these tables give a summary of the characteristics of the instances: number of nodes and required edges, a lower bound on the number of vehicles required, K^*, and the best upper bound known, UB. The superscripts on each upper bound indicate the CARP heuristics that obtained it:

a: Belenguer et al. [23], b: Hertz et al. [24],
c: Pearn [25] and d: Lacomme et al. [26].

Table 1
Instances from Benavent et al. [7]

Inst.	$\|V\|$	$\|R\|$	K^*	UB	LB2	CPA1	CPA2	Gap saving	#iter	#cuts (1,2)	#cuts (4,5,9)	CPU time CPA1	CPU time CPA2
1.A	24	39	2	247[a,b,d]	247	247	247		5	25	0	0.46	0.47
1.B	24	39	3	247[a,b,d]	247	247	247		7	26	0	0.56	0.56
1.C	24	39	8	319[b,d]	280	309	309	0	15	44	0	0.67	0.71
2.A	24	34	2	298[a,b,d]	296	298	298		3	21	0	0.13	0.13
2.B	24	34	3	330[a,b,d]	318	328	330	100	4	22	1	0.20	0.24
2.C	24	34	8	528[b,d]	482	526	526	0	7	33	0	0.38	0.40
3.A	24	35	2	105[a,b,d]	103	105	105		3	25	0	0.16	0.18
3.B	24	35	3	111[a,b,d]	108	111	111		3	27	0	0.14	0.15
3.C	24	35	7	162[a,b,d]	140	159	161	66.67	17	57	2	0.70	0.91
4.A	41	69	3	522[a,b,d]	514	522	522		4	37	0	0.54	0.54
4.B	41	69	4	534[a,b,d]	518	534	534		4	38	0	0.29	0.29
4.C	41	69	5	550[a,d]	524	550	550		7	44	0	0.94	0.97
4.D	41	69	9	652[d]	565	640	644	33.33	27	93	11	2.17	7.16
5.A	34	65	3	566[a,b,d]	562	566	566		10	38	0	1.36	1.36
5.B	34	65	4	589[a,b,d]	567	586	589	100	13	47	1	1.93	2.13
5.C	34	65	5	617[a,b,d]	582	610	612	28.57	11	46	1	1.46	1.70
5.D	34	65	9	724[d]	656	714	714	0	9	50	0	0.82	0.95
6.A	31	50	3	330[a,b,d]	330	330	330		5	31	0	0.60	0.62
6.B	31	50	4	340[a,d]	334	336	338	50	9	35	3	0.63	2.30
6.C	31	50	10	424[b,d]	372	414	418	40	17	62	3	0.95	1.80
7.A	40	66	3	382[a,b,d]	382	382	382		2	29	0	0.33	0.34
7.B	40	66	4	386[a,b,d]	382	386	386		3	29	0	0.18	0.20
7.C	40	66	9	437[a,b,d]	403	430	436	85.71	16	61	5	1.35	3.16
8.A	30	63	3	522[a,b,d]	522	522	522		3	22	0	0.33	0.34
8.B	30	63	4	531[a,b,d]	528	531	531		2	22	0	0.12	0.12
8.C	30	63	9	663[d]	587	645	653	44.44	25	61	7	0.91	3.73
9.A	50	92	3	450[a,b,d]	450	450	450		12	57	0	4.29	4.32
9.B	50	92	4	453[a,b,d]	453	453	453		6	47	0	1.97	1.97
9.C	50	92	5	459[a,b,d]	459	459	459		6	47	0	2.07	2.07
9.D	50	92	10	518[d]	493	505	509	30.77	32	92	11	2.32	21.84
10.A	50	97	3	637[a,b,d]	637	637	637		5	47	0	1.66	1.67
10.B	50	97	4	645[a,b,d]	641	645	645		5	48	0	1.42	1.44
10.C	50	97	5	655[a,d]	649	655	655		5	50	0	0.81	0.83
10.D	50	97	10	739[d]	697	731	732	12.50	24	108	7	3.80	8.38
Average gap %					3.90	0.66	0.41						
Maximum gap %					13.58	3.13	3.13						
Number of optima					9	20	22						

LB2: Benavent et al. [7], CPA1, CPA2: Cutting Plane Algorithm. *Note*: a–d explained in the text.

The number of vehicles used in the corresponding heuristic solutions is not always equal to the minimum possible, K^*.

It is important to remark that the upper bounds reported in [24,26] for the instances of Benavent et al. [7] differ from those we show in Table 1. The reason is that they use a different cost of

Table 2
Instances from Golden et al. [20]

Inst.	$\|V\|$	$\|R\|$	K^*	UB	LB2	CPA2	#iter	#cuts (1,2)	CPU time
gdb1	12	22	5	316[a,b,c,d]	310	316	4	14	0.11
gdb2	12	26	6	339[a,b,d]	339	339	2	10	0.08
gdb3	12	22	5	275[a,b,c,d]	275	275	2	11	0.1
gdb4	11	19	4	287[a,b,c,d]	274	287	2	9	0.09
gdb5	13	26	6	377[a,b,d]	376	377	5	14	0.15
gdb6	12	22	5	298[a,b,d]	295	298	2	9	0.08
gdb7	12	22	5	325[a,b,c,d]	312	325	3	11	0.14
gdb8	27	46	10	348[b,d]	326	344	6	42	0.27
gdb9	27	51	10	303[d]	277	303	8	40	0.42
gdb10	12	25	4	275[a,b,c,d]	275	275	2	8	0.08
gdb11	22	45	5	395[a,b,d]	395	395	3	23	0.54
gdb12	13	23	7	458[a,b,d]	428	450	3	12	0.11
gdb13	10	28	6	538[d]	536	536	2	7	0.16
gdb14	7	21	5	100[a,b,c,d]	100	100	1	1	0.05
gdb15	7	21	4	58[a,b,c,d]	58	58	1	1	0.04
gdb16	8	28	5	127[a,b,c,d]	127	127	1	8	0.08
gdb17	8	28	5	91[a,b,c,d]	91	91	1	8	0.04
gdb18	9	36	5	164[a,b,c,d]	164	164	1	1	0.06
gdb19	8	11	3	55[a,b,c,d]	55	55	1	7	0.04
gdb20	11	22	4	121[a,b,d]	121	121	3	9	0.13
gdb21	11	33	6	156[a,b,c,d]	156	156	2	6	0.08
gdb22	11	44	8	200[a,b,c,d]	200	200	1	5	0.06
gdb23	11	55	10	233[c]	233	233	2	2	0.09
Average gap %					1.46	0.14			
Maximum gap %					8.58	1.75			
Number of optima					14	20			

LB2: Benavent et al. [7], CPA2: Cutting Plane Algorithm. *Note*: a–d explained in the text.

Table 3
Instances from Kiuchi et al. [3]

Inst.	$\|V\|$	$\|R\|$	K^*	UB	LB2	CPA2	#iter	#cuts (1,2)	CPU time
kshs1	8	15	4	14661[a,b]	14039	14661	2	7	0.05
kshs2	10	15	4	9863[a,b]	9275	9863	2	11	0.08
kshs3	6	15	4	9320[a,b]	9320	9320	1	6	0.05
kshs4	8	15	4	11498[b]	10774	11098	2	5	0.06
kshs5	8	15	3	10957[a,b]	10957	10957	2	7	0.05
kshs6	9	15	3	10197[a]	10197	10197	2	8	0.06
Average gap %					2.75	0.58			
Maximum gap %					6.30	3.48			
Number of optima					3	5			

LB2: Benavent et al. [7], CPA2: Cutting Plane Algorithm. *Note*: a–d explained in the text.

Table 4
Eglese instances [21,22]

Inst.	$\|V\|$	$\|R\|$	K^*	UB	LB2	CPA1	CPA2	Gap saving	#iter	#cuts (1,2)	#cuts (4,5,9)	CPU time CPA1	CPU time CPA2
egl-e1-A	77	51	5	3548[d]	3071	3498	3515	34.00	46	128	3	12.96	32.00
egl-e1-B	77	51	7	4498[d]	3684	4436	4436	0.00	53	205	0	32.38	35.82
egl-e1-C	77	51	10	5595[d]	4386	5449	5453	2.74	43	188	4	10.99	22.37
egl-e2-A	77	72	7	5018[d]	4164	4907	4994	78.38	61	183	8	14.32	50.51
egl-e2-B	77	72	10	6340[d]	5050	6249	6249	0.00	34	189	0	17.32	19.11
egl-e2-C	77	72	14	8415[d]	6112	8105	8114	2.90	37	206	2	11.95	18.78
egl-e3-A	77	87	8	5898[d]	4885	5857	5869	29.27	57	200	6	38.45	86.78
egl-e3-B	77	87	12	7822[d]	6065	7570	7646	30.16	59	253	16	28.36	58.25
egl-e3-C	77	87	17	10433[d]	7463	10011	10019	1.90	39	202	2	9.89	19.90
egl-e4-A	77	98	9	6461[d]	5409	6370	6372	2.20	33	163	4	18.54	43.08
egl-e4-B	77	98	14	9021[d]	6935	8807	8809	0.93	36	188	1	12.73	15.87
egl-e4-C	77	98	19	11779[d]	8684	11262	11276	2.71	38	189	2	11.08	16.54
egl-s1-A	140	75	7	5018[d]	3448	4975	4992	39.53	98	386	5	267.44	1085.52
egl-s1-B	140	75	10	6435[d]	4012	6180	6201	8.24	77	327	37	130.82	216.06
egl-s1-C	140	75	14	8518[d]	5212	8268	8310	16.80	68	272	15	18.04	102.70
egl-s2-A	140	147	14	9995[d]	6803	9718	9780	22.38	103	447	77	192.81	521.77
egl-s2-B	140	147	20	13174[d]	8377	12835	12886	15.04	93	456	17	90.79	170.19
egl-s2-C	140	147	27	16795[d]	10093	16216	16221	0.86	105	610	2	229.71	274.21
egl-s3-A	140	159	15	10296[d]	7127	9991	10025	11.15	82	358	32	153.45	206.29
egl-s3-B	140	159	22	14053[d]	8867	13520	13554	6.38	119	468	29	216.24	504.29
egl-s3-C	140	159	29	17297[d]	10665	16958	16969	3.24	103	512	15	143.44	182.69
egl-s4-A	140	190	19	12442[d]	8488	12007	12027	4.60	88	390	23	186.46	324.66
egl-s4-B	140	190	27	16531[d]	10516	15897	15933	5.68	93	396	15	219.2	1173.00
egl-s4-C	140	190	35	20832[d]	12682	20176	20179	0.46	97	521	2	194.41	210.98
Average gap %					28.37	2.58	2.40						
Maximum gap %					39.90	4.39	4.27						

LB2: Benavent et al. [7], CPA1, CPA2: Cutting Plane Algorithm. *Note*: a–d explained in the text.

servicing the required edges of these instances. Given that this a fixed cost incurred by any solution, the only consequence of that difference in the data is that a constant term should be added to their reported upper and lower bounds. Let us denote by Q_I, for $I = 1, \ldots, 10$, the constant to be added for instance IX, $X \in \{A, B, C, D\}$. The constant values are: $Q_1 = 74$, $Q_2 = 71$, $Q_3 = 24$, $Q_4 = 122$, $Q_5 = 143$, $Q_6 = 107$, $Q_7 = 103$, $Q_8 = 136$, $Q_9 = 127$, and $Q_{10} = 209$.

The next column of the tables shows the lower bound LB2 (Benavent et al. [7]) that seems to produce the best results from among the existing lower bounds for the CARP. The next two columns, entitled CPA1 and CPA2, respectively, show the lower bounds obtained by our cutting plane algorithm at different phases. CPA1 is the lower bound obtained in the first iteration where no violated capacity or odd set constraint was found; that is, CPA1 is the lower bound that would be obtained if only these types of constraints were used in the cutting plane. CPA2 is the lower bound obtained by the complete cutting plane algorithm. Thus, CPA2-CPA1 indicates the improvement in the lower

bound that is due to the disjoint paths constraints (4), (5) and (9). The next column shows the gap saving, computed as (CPA2-CPA1)*100/(UB-CPA1), that represents this improvement; obviously it is computed only for those instances where CPA1 is not already optimal. Optimal lower bounds are underlined. Finally, the remaining columns show: the number of iterations of the cutting plane algorithm, the number of constraints (cuts) included in the last LP and the CPU time in seconds. The constraints have been grouped into two sets: capacity and odd set constraints (1,2) and disjoint paths constraints (4,5,9). A summary of the results for the different lower bounds is given at the bottom of each table. For each instance and lower bound, we have computed its percentage deviation from the upper bound (gap) as (UB-lower bound)*100/UB.

Tables 2 and 3 do not contain the column corresponding to the cuts of types (4), (5) and (9) because no cut of these types was found in the corresponding instances (except for instance gdb12 where only one of such cuts was found). This can be explained by the special characteristics of these instances: the associated graphs are quite small and the paths from the depot to any node have a small number of edges.

It has to be remarked that Golden et al. [20] and Pearn [25] reported a lower bound for instance gdb13, computed with the method of Golden and Wong [1], equal to 544. Subsequently, Coutinho-Rodrigues et al. [27] pointed out that the correct value of this lower bound is 536.

6. Concluding remarks

The results show that the cutting plane algorithm performs quite well: the average gap is 0.41% for the instances from Benavent et al. [7], 0.14% for those from Golden et al. [20], 0.58% for the instances from Kiuchi et al. [3], and 2.40% for the large Eglese instances. The respective average gaps for the lower bound LB2 in these sets of instances are 3.90%, 1.46%, 2.75% and 28.37%. On the other hand, in 47 out of the 63 instances from the first three sets tested, the lower bound is equal to the upper bound, thus proving the optimality of the corresponding heuristic solutions. These include several instances of about 50 nodes and 97 required edges. Furthermore, in 7 additional instances the gap is less than 1%, so they could probably be solved with a Branch & Cut algorithm. Concerning the Eglese instances, no optimal lower bound was obtained and only in 4 of them is the gap less than 1%. Given that we do not know the optimal cost of some instances, it is not clear at this stage if a large gap of a particular instance is due to the poor quality of either the upper bound, the lower bound, or both. It also has to be remarked that the CPU time consumed to obtain the lower bound is rather small even for the largest instances, and the computer used is an old one which is several times slower than a current cheap personal computer.

Acknowledgements

We thank Richard Eglese for making available the data of the road network of the county of Lancashire. We also thank Michel Mittaz and Christian Prins for running for us their respective CARP heuristics with the new set of Eglese instances. Finally, we thank to two anonymous referees for their valuable comments.

References

[1] Golden BL, Wong R. Capacitated arc routing problems. Networks 1981;11:305–15.
[2] Hirabayashi R, Saruwatari Y, Nishida N. Tour construction algorithm for the capacitated arc routing problem. Asia–Pacific Journal of Operational Research 1992;9:155–75.
[3] Kiuchi M, Shinano Y, Hirabayashi R, Saruwatari Y. An exact algorithm for the capacitated arc routing problem using Parallel Branch and Bound method, Abstracts of the 1995 Spring National Conference of the Operational Research Society of Japan, 1995. p. 28–9 [in Japanese].
[4] Assad A, Pearn WL, Golden BL. The capacitated Chinese postman problem: lower bounds and solvable cases American Journal of Mathematical and Management Science 1987;7:63–88.
[5] Pearn WL. New lower bounds for the capacitated arc routing problems. Networks 1988;18:181–91.
[6] Zaw Win. Contributions to routing problems. Dissertation, Universität Augsburg, Germany, 1988.
[7] Benavent E, Campos V, Corberán A, Mota E. The capacitated arc routing problem. Lower bounds. Networks 1992;22:669–90.
[8] Assad A, Golden BL. Arc routing methods and applications. In: Ball MO et al., editors. Handbook on operations research and management science, vol. 8. Amsterdam: Elsevier, 1995. p.375–483.
[9] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, Part 1: The Chinese postman problem. Operations Research 1995;43:231–42.
[10] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, Part 2: The rural postman problem. Operations Research 1995;43:399–414.
[11] Dror M. (ed.), Arc routing. Theory, solutions and applications. Boston: Kluwer Academic Publishers, 2000.
[12] Eglese RW, Letchford A. Polyhedral theory for arc routing problems. In: Dror M, editor. Arc routing. Theory, solutions and applications. Boston: Kluwer Academic Publishers, 2000. p. 199–230.
[13] Benavent E, Corberán A, Sanchis JM. Linear programming based methods for solving arc routing problems. In: Dror M, editor. Arc routing. Theory, solutions and applications. Boston: Kluwer Academic Publishers, 2000. p. 231–75.
[14] Belenguer JM, Benavent E. Polyhedral results on the capacitated arc routing problem. Dep. Estadística e Inv. Op., T.R. 1-92, Univ. de Valencia, 1992.
[15] Belenguer JM, Benavent E. The capacitated arc routing problem: Valid inequalities and facets Computational Optimization and Applications 1998;10:165–87.
[16] Padberg MW, Rao MR. Odd minimum cut-sets and b-matchings. Mathematics of Operations Research 1982;7:67–80.
[17] Harche F, Rinaldi G. The capacity inequalities for the Capacitated Vehicle Routing Problem, 1993, private communication.
[18] Bertsekas DP, Tseng P. The relax codes for linear minimum cost network flow problems. Annals of Operations Research 1988;13:125–90.
[19] CPLEX Optimization Inc., CPLEX Version 3.0, 1994.
[20] Golden BL, DeArmon JS, Baker EK. Computational experiments with algorithms for a class of routing problems. Computers and Operations Research 1983;10:47–59.
[21] Li LYO. Vehicle Routing for Winter Gritting. Dissertation, Lancaster University, UK, 1992.
[22] Li LYO, Eglese RW. An Interactive Algorithm for Vehicle Routing for Winter-Gritting. Journal of the Operational Research Society 1996;47:217–28.
[23] Belenguer JM, Benavent E, Cognata F. Un Metaheurístico para el Problema de Rutas por Arcos con Capacidades, talk presented at the 23th National SEIO Meeting, 1997, Valencia, Spain.
[24] Hertz A, Laporte G, Mittaz M. A tabu search heuristic for the capacitated arc routing problem. Operations Research 2000;48:129–35.
[25] Pearn WL. Approximate solutions for the capacitated arc routing problem. Computers and Operations Research 1989;16:589–600.
[26] Lacomme P, Prins C, Ramdane-Chérif W. A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers EJW et al., editors. Applications of evolutionary computing. Lecture Notes in Computer Science, vol. 2037. Berlin: Springer, 2001. p. 473–83.
[27] Coutinho-Rodrigues J, Rodrigues N, Climaco J. Solving an urban routing problem using heuristics: a successful case study. Belgian Journal of Operations Research, Statistics and Computer Science 1993;33(1,2):19–32.

José M. Belenguer is an Assistant Professor in the Department of Statistics and Operations Research at the Universitat de Valéncia, Spain. His interests concern arc routing, vehicle routing, combinatorial optimization, exact and heuristic methods.

Enrique Benavent is an Assistant Professor in the Department of Statistics and Operations Research at the Universitat de Valéncia, Spain. His interests concern arc routing, vehicle routing and scheduling, combinatorial optimization, graphs, exact and heuristic methods.