The Capacitated Arc Routing Problem: Lower bounds

Article *in* Networks · December 1992

DOI: 10.1002/net.3230220706

4 authors:

Enrique Benavent
University of Valencia
33 PUBLICATIONS **2,163** CITATIONS
SEE PROFILE

Vicente Campos
University of Valencia
39 PUBLICATIONS **1,719** CITATIONS
SEE PROFILE

Ángel Corberán
University of Valencia
109 PUBLICATIONS **2,940** CITATIONS
SEE PROFILE

E. Mota
University of Valencia
13 PUBLICATIONS **535** CITATIONS
SEE PROFILE

The Capacitated Arc Routing Problem: Lower Bounds

E. Benavent, V. Campos, A. Corberan, and E. Mota
Departamento De Estadistica e Investigacion Operativa
Facultad de Matematicas
Universidad de Valencia
Dr. Moliner n. 50
46100 Burjassot-Valencia, Spain

In this paper, we consider the Capacitated Arc Routing Problem (CARP), in which a fleet of vehicles, based on a specified vertex (the depot) and with a known capacity Q, must service a subset of the edges of a graph, with minimum total cost and such that the load assigned to each vehicle does not exceed its capacity. New lower bounds are developed for this problem, producing at least as good results as the already existing ones. Three of the proposed lower bounds are obtained from the resolution of a minimum cost perfect matching problem. The fourth one takes into account the vehicle capacity and is computed using a dynamic programming algorithm. Computational results, in which these bounds are compared on a set of test problems, are included.

INTRODUCTION

Routing problems have been widely studied during the last years, mainly because of the great number of practical applications and the big increase of the costs associated with operating the vehicles. Basically, these problems can be divided into Node Routing Problems, if the demand occurs in the nodes or vertices of a graph, and Arc Routing Problems, in which the pickup or delivery activities occur along the arcs or edges of a graph. See [5] and [14] for a more detailed classification of these problems and their complexity. However, the research work has been focused mainly on Node Routing Problems (see the excellent survey [6]), while Arc Routing Problems have received comparatively little attention (see [3]) in spite of their applications in a great number of real problems, such as problems of refuse collection, street sweeping operations, delivery of milk or post, inspection of distributed systems (electric power, telephone, or railway lines), and school bus routing.

In this work, we consider the Capacitated Arc Routing Problem (CARP), in which a fleet of vehicles, based on a specified vertex (the depot) and with a known capacity Q, must service the edges with positive load of a graph, with minimum total cost and such that the load assigned to each vehicle does not exceed its capacity.

Although Assad et al. [1] showed that certain classes of the CARP with special graph, demand, or cost structures can be solved with a polynomial time algorithm, the general case is an NP-hard problem. Problems such as the Rural Postman Problem (RPP) and the Traveling Salesman Problem (TSP) are special cases of the CARP and even the problem of finding a solution with a cost less than 1.5 times the optimal solution cost is NP-hard, as shown by Golden and Wong [11].

Because of the complexity of the problem, several heuristic algorithms and lower bounding techniques have been developed for the CARP or for some special cases. Among the heuristics, we could mention those of Christofides [8], Beltrami and Bodin [2] for the routing of street sweepers, Male and Liebman [15] for a refuse collection problem, Stern and Dror [18] for the routing of electric meter readers, Chapleau et al. [7] for school bus routing, Golden et al. [12], Golden and Wong [11], and Benavent et al. [4].

In this article, new lower bounds are developed for the problem, producing at least as good results as the ones obtained by Golden and Wong [11], Assad et al. [1], Pearn [16], and Zaw Win [19], the only lower bounds published as far as we know. The article is organized as follows: The problem and some notation are introduced in Section 1. In Section 2, the existing lower bounding procedures for the CARP are described with special emphasis on the newest ones by Pearn [16] and Zaw Win [19]. Section 3 contains three new procedures, denoted LB1, LB2, and LB3, to compute lower bounds based on matching techniques. The third of them, LB3, applies to the case where the number of vehicles is fixed. It is also shown that LB1 dominates the Pearn bound and that LB2 dominates LB1 and Zaw Win bounds. In Section 4, we present a dynamic programming-based technique to compute lower bounds to the CARP with a fixed number of vehicles. Several improvements, including Lagrangean relaxation, have been developed for this technique, which has the desirable feature of producing not only lower bounds but almost feasible routes. Nevertheless, the bound produced with these techniques is not as good as the preceding ones. Besides their possible use in an exact algorithm, these lower bounds allow to measure the efficiency of the heuristics. Computational results, in which these bounds are compared on a set of test problems, are included in Section 5.

1. PROBLEM DEFINITION AND NOTATION

Let $G = (V, E)$ be a connected and undirected graph. For every edge $e \in E$, consider a load $q_e \geq 0$ and a traversing cost $c_e \geq 0$. Moreover, associated to every edge e with $q_e > 0$, a servicing cost $c'_e \geq c_e$ will be considered. Given vertex 1, representing the depot, and a fleet of vehicles of capacity $Q(Q \geq \max\{q_e\colon e \in E\})$, the CARP consists of finding a set of vehicle routes with minimum total cost, such that each route contains the depot, each edge with positive load is serviced exactly once by any vehicle, and the capacity of the vehicles is not exceeded. Note that a route is determined by a set of traversed edges, with indication of the serviced ones.

Edges e with $q_e > 0$ will be called required edges. Let E_R be the set of required edges, and V_R, the set of vertices including the depot and very vertex incident with at least one required edge. Let $G_R = (V_R, E_R)$ and $d(i)$ be the degree of vertex i in G_R.

Let C_T denote the sum of the servicing costs, c'_e, of all the edges in E_R and Q_T, the total load. Obviously, at least, $K_0 = \lceil Q_T/Q \rceil$ vehicles are needed to service all the edges. Let s_{ij} denote the cost of a shortest path in G from vertex i to vertex j using costs c_{ij}; costs s_{ij} will be also referred to as distances.

A feasible solution to the CARP consists of a set of routes that service, jointly, all the edges of G_R; each edge is serviced only once though some edges can be traversed more than once. The *augmented graph* associated to a feasible solution is obtained by adding to $G - \{e \in E: q_e = 0\}$ (the graph obtained from G deleting the edges with zero load) as many copies of each edge as the number of times this edge has been traversed without being serviced by the set of routes. The added edges will be referred to as *artificial edges*.

A vertex is even in a given graph if its degree is even. A graph is called even if all its vertices are even. For any subset of vertices V' of the set $V - \{1\}$, let $\delta(V') = \{e = (i,j) \in E: i \in V', j \in V - V'\}$; $\delta(V')$ will be called an edge cutset. Given a graph H with an even number of vertices and costs associated to its edges, MP(H) will denote the cost of the Minimum Cost Perfect Matching (MCPM) defined on H.

2. EXISTING LOWER BOUNDS

Let G^* denote the augmented graph associated to an optimal solution of the CARP. Obviously, this graph is even and the degree of the depot in it is, at least, $2K_0$; furthermore, there exists in G^* a set of paths, formed by artificial edges, such that every odd vertex of G_R is a terminal vertex of at least one of these paths. Let S be the set of odd vertices of G_R and let H be the complete graph with set of vertices S and costs s_{ij}. If $d(1) \geq 2K_0$, then $C_T + \mathrm{MP}(H)$ is a lower bound to the optimal value of the CARP, because MP(H) corresponds to the minimum cost of making even the graph G_R by adding artificial edges.

If $d(1) < 2K_0$, an improved lower bound can be obtained. Let $J = 2K_0 - d(1) > 0$. Obviously, at least J artificial edges will be incident to the depot in G^*. Golden and Wong [11] proposed to compute an MCPM on a complete graph H' whose set of vertices includes not only the set S but two sets A and B, each one with J vertices ($J - 1$ vertices if $1 \in S$). Vertices in A can be considered as copies of the depot and they are included to fulfill the requirement that at least J artificial edges must be incident with the depot. The costs of the edges of H' are s_{ij} for the edges with endpoints $i, j \in S$, s_{1i} for the edges between $i \in S$ and any vertex in A; c^* (the cost of the least cost edge incident with the depot) for the edges between A and B; zero for the edges with both endpoints in B and infinite for all the other edges. Then, C_T + MP(H') is a lower bound for the CARP.

Another procedure was proposed by Assad et al. [1]. If the vertices in G_R are numbered in nondecreasing order with respect to their distances to the depot ($s_{12} \leq s_{13} \leq \cdots$), i^* is the smallest integer satisfying $d(2) + d(3) + \cdots + d(i^*) \geq J$ and $d(i^*)$ is redefined as $d(i^*) = J - (d(2) + \cdots + d(i^* - 1))$, then $C_T + s_{12}d(2) + \cdots + s_{1i^*}d(i^*)$ is a lower bound for the CARP. This bound is based on the idea that if $J > 0$ some vehicles will traverse a path from the depot to a given vertex before servicing any edge; obviously, the number of paths ending in a given vertex cannot be greater than the number of required edges incident to it.

Pearn [16] proposed a two-stage iterative procedure that combines the above two methods and that is shown to dominate the Golden and Wong bound.

Pearn Procedure

Stage 1. Given an even number p, $0 \leq p \leq |S|$, an MCPM is computed on the set $S \cup A(p)$, where $A(p)$ contains p copies of the depot [it is assumed that $d(1)$ is even]. Edge costs correspond to those of the shortest paths in G, except edges between vertices in $A(p)$, which have cost infinite. Let MP($S \cup A(p)$) be the cost of this matching. Edges in the original graph corresponding to the above matching solution are added, thus producing a new graph. Let $d'(1)$ be the degree of the depot in this graph.

Stage 2. In the second stage, all the vertices in G_R are renumbered so that $s_{12} \leq s_{13} \leq \cdots \leq s_{1n}$, where $n = |V_R|$. Let $r = \min\{j: d(2) + \cdots + d(j) \geq 2K_0 - d'(1)\}$ and update

$$d(r) = 2K_0 - d'(1) - \sum_{j=2}^{r-1} d(j).$$

Compute LB(p) = C_T + MP($S \cup A(p)$) + $\Sigma_{i=2}^{r} s_{1i}d(i)$. The Pearn lower bound is given by PEARN = min{LB(p), p = 0,2,4, . . . , $|S|$}.

Recently, Zaw Win [19] proposed a number of lower-bounding procedures containing new interesting ideas. In particular, he considered successive edge cutsets instead of considering only the edge cutset $\delta(\{1\})$ as in the previous procedures. We describe here only the two lower bounds of Zaw Win that we consider most promising.

Procedure for ZAW1 and ZAW2

```
Step 1. Set $U = \{1\}$, $L1 = 0$, $L = 0$, $L2 = 0$.
Step 2. Let $V' = V - U$ and $G'$ be the graph induced by $V'$. Find the
        connected components of $G'$. Suppose that $G'$ has $t$ components
        $G'_s = (V'_s, E'_s)$, $1 \leq s \leq t$.
    (i) For $s$ = 1 to $t$ do:
        (i.1)
```

$$p_s = \left\lceil \frac{\sum_{e \in E'_s \cup \delta(V'_s)} q_e}{Q} \right\rceil \qquad q_s = |\{e \in \delta(V'_s) : q_e > 0\}|$$

$$r_s = 2\, p_s - q_s$$

$$\bar{c}_s = \min_{e \in \delta(V'_s)} c_e$$

If $r_s < 0$, set $r_s = 0$ if q_s is even or set $r_s = 1$ if q_s is odd.

Set $m_s = \bar{c}_s r_s$

(i.2) Let $S'_s = S \cap V'_s$. If $S'_s \neq \varnothing$, construct a weighted graph H_s (see below) and set $m_s = \mathrm{MP}(H_s)$.

(ii) $L = \sum_{s=1}^{t} m_s$

(iii) $L2 = \max\{L2, C_T + L + L1\}$

(iv) $L1 = L1 + \sum_{s=1}^{t} r_s \bar{c}_s$

Step 3. Set $U' = \{i \in V : i$ is adjacent to a vertex in $U\}$ and $U = U \cup U'$. If $U \neq V$ go to 2; otherwise, go to 4.

Step 4. Set ZAW1 $= L1 + C_T$ and ZAW2 $= L2$, stop.

This algorithm computes two valid lower bounds, denoted by ZAW1 and ZAW2, for the CARP. The procedure for computing only ZAW1 is obtained by ignoring (i.2) in Step 2. To compute this bound, the algorithm considers a sequence of edge cutsets $\delta(U)$, pairwise disjoints, and computes, for each one, a lower bound on the cost of the artificial edges, in the edge cutset, that would be used by any feasible solution. The sum of these costs, for each edge cutset $\delta(U)$, is then a valid lower bound for the CARP. Note that the graph induced by $V - U$ may have several connected components, so $\delta(U)$ can be decomposed in several edge cutsets $\delta(V'_s)$. For each connected component, p_s is the minimum number of vehicles needed to service the required edges in G'_s and in $\delta(V'_s)$ and, then at least r_s artificial edges of $\delta(V'_s)$ will be used by any feasible solution.

Consider now the whole algorithm, i.e., including (i.2) in Step 2. For each edge cutset $\delta(U)$, the algorithm computes a lower bound, given by $C_T + L + L1$ and ZAW2 is the maximum for all the edge cutsets [Step 2 (iii)]. Note that $L1$ is a lower bound on the cost of the artificial edges used by any feasible solution in the subgraph induced by U. On the other hand, L is a lower bound on the costs of the artificial edges in the remaining graph, as it can be seen from the construction of graphs H_s.

Graph H_s is a complete weighted graph whose vertex set is $S'_s \cup A \cup B \cup X \cup Y$, where $S'_s = S \cap V'_s$ and A, B, X, Y are sets of artificial vertices such that

$|A| = |B| = r'_s$, where $r'_s = r_s$ if $|S'_s|$ is even and $r'_s = r_s - 1$ otherwise.

$|X| = \max\{0, |S'_s| - r_s\}$ and

Y is empty if $|S'_s|$ is even and contains a single vertex otherwise.

Note that q_s (the number of required edges in the cut set) and $|S'_s|$ (the number of vertices of odd degree with respect to the required edges in G'_s) have the same parity and, therefore, H_s always contains an even number of vertices.

Costs of the edges in H_s are infinite except for the following costs:

$$c_{uv} = \begin{cases} s_{uv} \text{ if } u, v \in S'_s \\ \text{minimum distance between } u \text{ and the vertices of } U \text{ if } u \in S'_s \\ \quad \text{and } v \in A \cup X \cup Y \\ \bar{c}_s \text{ if } u \in A \text{ and } v \in B \\ 0 \text{ if } u, v \in B \text{ or } u, v \in X. \end{cases}$$

Note that bound ZAW2 generalizes the bound of Golden and Wong by considering successive edge cutsets. In particular, it is easy to see that in the first iteration, when $U = \{1\}$, ZAW2 equals the bound of Golden and Wong.

3. MATCHING-BASED LOWER BOUNDS

In this section, three methods of computing lower bounds to the CARP are described, all of them require the resolution of an MCPM problem. These methods can be considered as improvements of the ones proposed by Golden and Wong [11], Assad et al. [1] and Zaw Win [19].

3.1. Bound LB1

Assume that the vertices in G_R are $1, 2, \ldots, |V_R|$, numbered in nondecreasing order with respect to their distances to the depot, so $s_{12} \leq s_{13} \leq \ldots$ and construct a complete graph $G^a = (V^a, E^a)$ with $V^a = A \cup B \cup S'$, where

$A = \{a_1, \ldots, a_J\}$ is a set of copies of the depot,
B contains $d(i)$ copies of vertex i, for $i = 2, \ldots, r$, where r is the minimum value of p such that $d(2) + \cdots + d(p) \geq J$ and
S' contains a copy of each vertex in S excluding the depot and those odd vertices whose copies are already included in B.

Costs of the edges in E^a are defined as follows: cost infinity between every pair of vertices of A, and, for all the other pairs, the cost of the shortest path in G between the corresponding vertices of G_R. Note that the costs of the edges

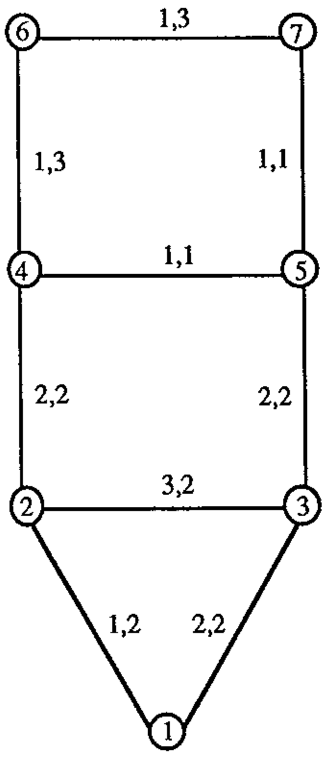

FIG. 1. The original graph.

between copies of the same vertex will be zero except for the copies of the depot.

The following theorem shows that $\mathrm{MP}(G^a) + C_T$ is a lower bound for the CARP and its proof is illustrated in Figures 1–4. Figure 1 represents the original graph, with the indicated traversing cost and load for each edge. The capac-

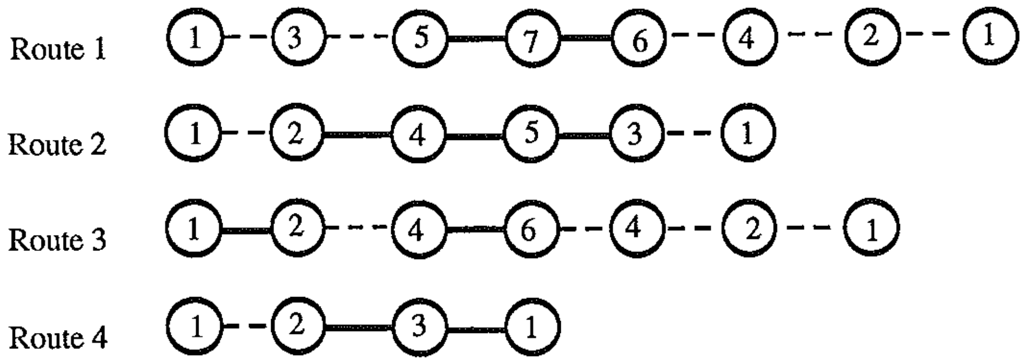

FIG. 2. The four routes in an optimal solution (---) traversed edge; (——) serviced edge.

ity of the vehicles is $Q = 5$. An optimal solution to the corresponding CARP consists of the four routes in Figure 2.

Theorem 1. A lower bound to the optimal value of the CARP is given by $\mathrm{LB1} = \mathrm{MP}(G^a) + C_T$.

Proof. Let G^* denote, as before, the augmented graph associated to an optimal solution of the CARP and let T be the set of artificial edges in G^*. Let $G(T) = (V, T)$ be the subgraph of G^* containing only these artificial edges. A perfect matching on G^a will be constructed with cost not greater than $c(T)$, the total cost of the edges in T, thus proving the theorem. To do this, at each step, a path in $G(T)$ will be identified and two vertices in G^a, corresponding to the terminal vertices of this path, will be matched with cost not greater than the cost of the path. The edges in this path will be then removed from $G(T)$. All along the procedure, $G(T)$ will denote the remaining graph after these eliminations. Also, the following property will always be true:

> The degree of a vertex in $G(T)$ is even if and only if there is an even number of unmatched copies of this vertex in G^a (zero will be considered an even number). (1)

Obviously, (1) is satisfied when the matching is empty. To construct the perfect matching on G^a, we first match the copies of the depot:

(a) Matching the copies of the depot (set A).

A *d-path* is defined as a path in $G(T)$ formed by the edges that have been traversed in the optimal solution by a given vehicle, from its departure point at the depot to the first serviced edge (or from the last serviced edge back to the depot). More generally, two d-paths can be identified each time a vehicle passes through the depot. We will consider as d-paths only those with at least one edge. Obviously, the number of d-paths is, at least, J, and, furthermore, the number of d-paths having $i \neq 1$ as terminal vertex is not greater than $d(i)$ because after (before) the d-path is traversed one edge incident with i will be serviced and edges are serviced only once. Let T' be the set of edges in T that belong to d-paths. Lines in Figure 3 join the terminal vertices of the six d-paths for the example; edge (2,4) is the only edge not in the set T'.

Apply the following procedures until all vertices of A (except possibly one) are matched:

(i) Identify in $G(T)$ a d-path having as terminal vertex one such that there exists at least one unmatched copy of this vertex in G^a, or a d-path that can be extended, by using edges in $G(T) - T'$ [the graph $G(T)$ deleting the edges in T'], to a path with a terminal vertex of the mentioned type. Match a vertex of A with a copy of this vertex and eliminate from $G(T)$ all the edges in the path. Repeat this procedure until such a path does not exist any longer (at this step, d-paths 1–5, 1–2, 1–3, and 1–2 in the example have been identified and four vertices in A are matched with,

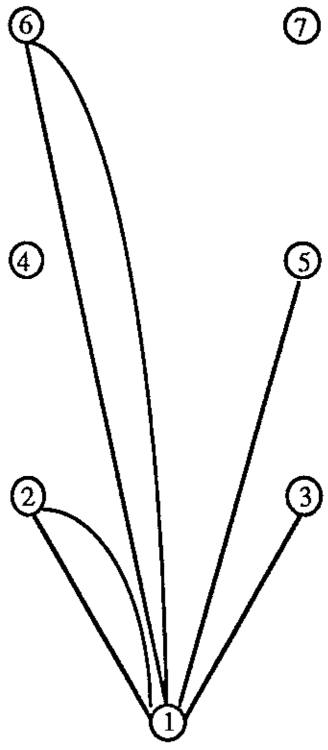

FIG. 3. The d-paths.

respectively, vertex 5, a copy of vertex 3, and two copies of vertex 2). From the definition of G^a, after applying this procedure, any d-path remaining in $G(T)$ has a cost not less than the cost of any edge in G^a joining an unmatched vertex of A with an unmatched vertex of B.

If at least two vertices of A remain unmatched, then

(ii) Identify a d-path in $G(T)$ and let v be the terminal vertex, which is obviously of even degree in $G(T)$ (there is no unmatched copy of v in G^a). Therefore, a path in $G(T) - T'$ can be found from v to a vertex v', which is also a terminal vertex of another d-path. Let c be the minimum cost of these two d-paths. So we have found a cycle of cost at least $2c$ (in our example, the cycle corresponds to the two d-paths having vertex 6 as terminal vertex). Then, two possibilities arise:

• There exist two unmatched copies of the same vertex in B (as it occurs with vertex 3 in the example). Then, match these two copies with two vertices of A and eliminate from $G(T)$ all the edges in the above-mentioned cycle.
• Otherwise, let $b_j \in B$ be an unmatched vertex in G^a corresponding to a vertex j in $G(T)$; b_j is the only unmatched copy of j in G^a and then j is odd

in $G(T)$. Therefore, there exists a path in $G(T) - T'$ from j to another odd-degree vertex j' in $G(T)$ (which, therefore, has an unmatched copy, say $v_{j'}$, in G^a). Let c' be the cost of this path. Then, eliminate from $G(T)$ the edges in the above-mentioned cycle and those in the path from j to j' (with total cost at least $2c + c'$) and match in G^a b_j and $v_{j'}$, with any pair of vertices in A. Note that the cost of matching b_j is not greater than c and that of $v_{j'}$ does not exceed $c + c'$ (the cost of a path from the depot to j' through vertex j).

Given that the number of d-paths in the original $G(T)$ was at least J and $|B| \geq J$, this procedure can be applied until all the vertices in A, except possibly one, have been matched.

We then complete the perfect matching as follows:

(b) Matching the remaining vertices.

Consider the odd degree vertices in $G(T)$. Beginning with one such a vertex, a path in $G(T)$ can be found (without taking into account whether the edges in the path belong, or not, to d-paths) that reaches another odd degree vertex in $G(T)$. Eliminate the edges in the path and match in G^a an unmatched copy of each of the terminal vertices of the path [these copies always exist because of (1)]. Repeat the procedure until there are no odd degree vertices in $G(T)$. If vertices in B, which represent copies of a given vertex in G, are unmatched at the end of the procedure, then match pairs of copies with zero cost. As the number of unmatched copies of any vertex in G is even (possibly zero), a perfect matching on G^a can be obtained. In our example, the odd degree vertices in $G(T)$ are vertices 2 and 4, connected by the edge (2,4). The perfect matching so constructed is shown in Figure 4 and its total cost is $14 + C_T$ (note that the MCPM on graph G^a has cost $10 + C_T$ [see Fig. 5] and this would be the value of LB1 for this example). ■

The lower bound of Golden and Wong [11] consists of computing an MCPM in a graph similar to G^a that differs basically in the costs of the edges incident with the copies of the depot, which are set equal to that of the shortest distance from the depot to any vertex in $V_R - \{1\}$. Thus, it is not hard to show that bound LB1 is always at least as good as the one of Golden and Wong.

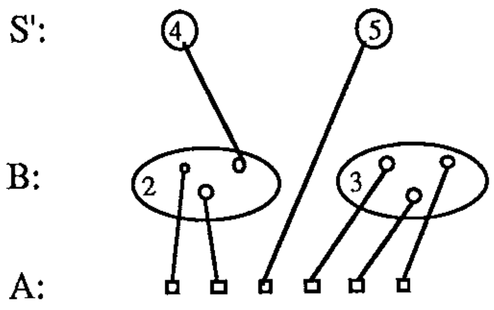

FIG. 4. The constructed perfect matching on G^a.

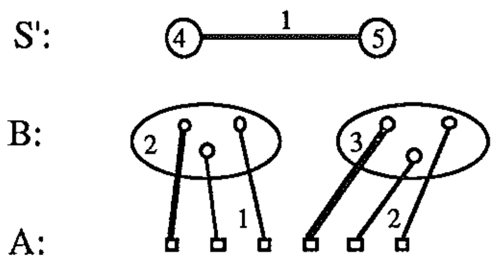

FIG. 5.

On the other hand, the lower bound of Assad et al. [1] only takes into account the costs of the J least-cost d-paths present in any feasible solution to the CARP, which correspond to the costs of the edges incident with the first J vertices of B in G^a. Therefore, this bound cannot be better than bound LB1. From this discussion, it is clear that bound LB1 combines the most significant features of the above lower bounds. The following theorem shows that LB1 dominates also the lower bound of Pearn [16].

Theorem 2. For every instance of the CARP, there exists an even number p, $0 \leq p \leq |S|$ such that LB1 $\geq$ LB(p), so LB1 $\geq$ PEARN.

Proof. Let us assume, for the sake of simplicity, that $d(1)$ is even. Let MP the optimal matching on G^a computed to obtain LB1. Suppose that, in MP, the copies of any even vertex of G are matched between them or with vertices in A (not with copies of an odd vertex). It is possible to modify any optimal matching on G^a to satisfy this condition without increasing the cost. Let S^1 be the set of odd-degree vertices $i \in S$ represented by $d(i)$ copies in B such that all their copies in G^a have been matched, in MP, between them or with vertices in A. Let p_1 be their number. Now, let S^2 be the set of vertices in S' matched, in MP, in G^a with vertices in A and let p_2 be their number. Define $p^* = p_1 + p_2$; it is easy to see that p^* is even and $0 \leq p^* \leq |S|$. Then, it is possible to construct a feasible solution to the MCPM problem on $S \cup A(p^*)$ (first stage of Pearn procedure) matching the vertices in $S^1 \cup S^2$ with the p^* copies of the depot and the remaining odd degree vertices in the same way as they have been matched in G^a. Then, the matching computed in the first stage of the Pearn procedure has cost not greater than a part of the perfect matching on G^a (bold lines in Fig. 5) computed to obtain LB1. Let MP' be the set of remaining edges of the perfect matching in G^a.

In the second stage of the Pearn procedure, the degree of the depot is completed in the cheapest way by using, if necessary, $d(2)$ times edge (1,2), $d(3)$ times edge (1,3), and so on. However, in the computation of bound LB1, we take into account that some of the copies of the closest vertices to the depot (those corresponding to the vertices in S^1) have been already used and, therefore, the cost of the edges in MP' may be greater (never less) than the cost of the second stage of the Pearn procedure. ■

The following example illustrates the proof outlined above.

Example. Consider the example of Figure 1. As before $Q = 5$, then $K_0 = 4$ and $J = 6$. Figure 5 shows an optimum matching in G^a computed to obtain LB1. Thus, LB1 $= 10 + C_T$.

From this solution, $p^* = 2$ and the matching formed by the edges (4,5), $(2,a_1)$, and $(3,a_2)$, indicated by bold lines, is a feasible solution to the first stage of the Pearn procedure with $p = 2$. At the second stage, four edges must be incident with the depot; thus, edge (1,2) is added three times and edge (1,3) only once and, therefore, LB(2) $= 4 + 5 + C_T$.

3.2. Bound LB2

Bound LB1 can be improved in the same way that ZAW2 improves the Golden and Wong bound by considering successive edge cutsets. The new lower bound will be called LB2 and it is best described by modifying the algorithm that produces ZAW2 presented in Section 2 (we follow the notation introduced for that algorithm unless otherwise stated). The main difference is in the way the graphs H_s are constructed although the following minor modifications are also necessary: First, in part (i.1) of Step 2, the computation of r_s is substituted by simply $r_s = \max\{0, 2p_s - q_s\}$ and m_s is set to zero. On the other hand, part (i.2) of the same step is substituted by

(i.2) Let $S'_s = S \cap V'_s$. If $S'_s \neq \emptyset$ or $r_s > 0$, construct the weighted graph H_s (see below) and set $m_s = \text{MCPM}(H_s)$.

The graph H_s is constructed as follows: Let us denote by $m(i)$ the minimum distance between vertex i in V'_s and any vertex of U and suppose that the vertices in V'_s are renumbered $i_1, i_2, \ldots$ in such a way that $m(i_1) \leq m(i_2) \leq \ldots$. Let h be the minimum integer such that $d(i_1) + \cdots + d(i_h) \geq r_s$. Then, the complete graph H_s will have as set of vertices $A \cup B \cup S''_s \cup X$, where

A is a set of r_s artificial vertices;
B contains $d(i_j)$ copies of vertex $i_j \in V'_s$ for $j = 1, \ldots, h$;
S''_s contains a copy of each vertex in S'_s except for those whose copies are already included in B;
X is a set of $\max\{0, |S'_s| - r_s\}$ artificial vertices.

Costs of the edges in H_s are infinite except for the following ones: for the edges between vertices in $B \cup S''_s$, the cost of the shortest path in G between the corresponding vertices of G_R (note that the costs of the edges between copies of the same vertex will be zero); for the edges between $u \in B \cup S''_s$ and $v \in A \cup X$, cost $m(j_u)$, where j_u denotes the corresponding vertex to u in G; and, finally, zero cost for the edges between vertices in X.

A proof similar to that of Theorem 1 would prove that MCPM(H_s) is a lower bound on the cost of the artificial edges in $V'_s \cup \delta(V'_s)$ so, LB2, computed as

ZAW2 except for the indicated modifications, is a lower bound for the CARP. Furthermore, it is obvious that, as LB1 dominates the bound of Golden and Wong, LB2 dominates ZAW2 and LB1.

3.3. Bound LB3

We consider here the special case of the CARP in which the number of available vehicles K equals the minimum number K_0.

Bound LB3 results from a more careful examination of the number of d-paths having the same terminal vertex. Let $Q_{\min} = Q_T - (K_0 - 1)Q$ (at least $Q_{\min}$ will be load by any vehicle in a feasible solution to the CARP). As an example, consider the graph in Figure 1. At first sight, three d-paths could be incident with vertex 2 (one for each incident edge), but a vehicle traveling without loading to vertex 2 and servicing edge (2,1), given that the load of this edge is less than $Q_{\min}$, would have to leave the depot once more generating another d-path. Therefore, we can suppose that only two d-paths incident with vertex 2 may be used. In what follows, we generalize this idea.

Given a vertex $i \in V_R$, let $\delta(i) = \{e \in E: e \text{ is incident with } i\}$. Let $e = (i,j)$ be an edge of E_R; we define $R(j)$ as the set of vertices of G that can be reached from vertex j using paths in $G\text{-}\delta(i)$ such that the depot is not an internal vertex of any of these paths. Let $R(i,e) = R(j) \cup \{i\}$ for each $e = (i,j) \in E_R$ and $G[R(i,e)]$, the subgraph of G induced by the set of vertices $R(i,e)$; we denote by $Q(R(i,e))$ its total load.

Suppose that the following conditions are satisfied for a vertex $i \in V_R$ and an edge $e = (i,j) \in E_R$:

(1) The depot is in $G[R(i,e)]$.
(2) $Q(R(i,e)) < Q_{\min}$.
(3) The shortest path in G from 1 to i traverses edge (i,j).
(4) Edge e is the only edge in G from i to any other vertex in $R(i,e) - \{1\}$.

If in a solution to the CARP a vehicle goes from 1 to i without servicing any edge and immediately services edge (i,j), to leave the subgraph $G[R(i,e)]$, the vehicle must return to the depot or traverse once more edge (i,j) from j to i. This last possibility, considering that (3) is satisfied, cannot occur in an optimal solution because the route could be improved by eliminating the two extra times edge (i,j) is traversed. Then, after the vehicle reaches the depot, it will have to leave it again for its load to be, at least, $Q_{\min}$. Therefore, following the argument given in the example, the number of d-paths incident with i can be supposed to be at most $d(i) - 1$.

Bound LB3 is as follows: From graph G_R, construct the complete graph $G^d = (V^d, E^d)$, where the set of vertices is $V^d = A \cup B'$. $A = \{a_1, \ldots, a_J\}$ represents the set of copies of the depot and B' contains $d(i)$ copies of each vertex $i \neq 1$ in G_R. Initially, costs of the edges in E^d are that of the shortest paths in G between the corresponding vertices, except for the copies of the depot, which are linked by edges of infinite cost. It is easy to show that the MCPM on G^d has the same cost as the one defined on G^a, the graph corresponding to bound LB1. To compute bound LB3, some costs of G^d will become infinite, using the following procedure:

Assume that the vertices are indexed as in Section 3.1. Beginning with $i = 2, 3, \ldots$, look for an edge (i,j) satisfying (1), (2), (3), and (4) above. If such an edge (i,j) exists, make infinite the cost of the edges among the J copies of the depot and one of the copies of i. Update $Q_{\min}$ to $Q_{\min} - Q(R(i,e))$ and continue until $Q_{\min} = 0$ or all the vertices are searched.

In the computational results given in Section 4, we have limited the search to the vertices i such that there exists in G a path of cardinality at most two between the depot and vertex i. The proof of the following theorem goes along the same lines as the proof of bound LB1.

Theorem 3. Let MP(G^d) be the cost of the MCPM on the resulting graph G^d. Then, LB3 = MP(G^d) + C_T is a lower bound to the optimal value of the CARP.

Obviously, LB3 $\geq$ LB1 and the computational results show that, in some cases, LB3 performs much better than LB1 (in the example of Figure 1, LB3 would be $14 + C_T$), although at a greater computational effort, since it involves the resolution of an MCPM on a graph with a number of vertices that is twice the number of required edges in the original graph.

In a recent paper [17], Saruwatari et al. propose a lower bound for the CARP, called NDLB, which involves the construction of a graph similar to the initial graph G^d used to compute LB3. It can be easily shown that NDLB equals LB1 although it involves a greater computational effort (similar to that of LB3). An improvement of NDLB is proposed in the same paper but it only applies when there exists, at least, a pair of edges whose total load is greater than the capacity of the vehicles. This situation does not appear in any of the instances tested in Section 5, so, for these instances, the bounds proposed in [17] equal LB1.

4. DYNAMIC PROGRAMMING-BASED LOWER BOUNDS

Bounds LB1, LB2, and LB3 do not take into account the constraints on the capacity of the vehicles, but indirectly and in a very limited way. However, these constraints can influence decisively the cost of the optimal solution, especially when the total capacity of the vehicles is close to the total load in the graph. In this section, we consider the special case of the CARP where the maximum number of vehicles to be used, $K \geq K_0$, is known in advance. Without loss of generality, we may also assume that the loads are integer numbers. We present lower bounds for this case that incorporate, directly, the capacity constraints of the vehicles and produce "routes," not always possible ones, for each vehicle.

4.1. Lower Bound to the Cost of a Route

In what follows, we present a dynamic programming algorithm producing a lower bound to the cost of a route from the depot to a given vertex, with given total load. The requirement that each required edge has to be serviced only once is relaxed in this bound.

Let a *q-chain* be defined as an ordered set of vertices and edges v_0, e_1, v_1, . . . , v_t, where e_i is incident to v_{i-1} and v_i, for $i = 1, \ldots, t$, together with a set of indices $D = \{i_1, \ldots, i_p\}$, indicating that edge e_i with $i \in D$ has been serviced while the remaining edges have only been traversed. The load of the q-chain is defined as the sum of the loads of the edges indexed in D (note that a given required edge may be serviced more than once). Let us define $V'_R = V_R - \{1\}$ if $d(1) = 0$ and $V'_R = V_R$ otherwise.

Let $u(j,q)$ be the cost of the least-cost q-chain from the depot to vertex $j \in V_R$ with a total load of q. These costs can be computed for every vertex $j \in V_R$ and any load $q = 0, \ldots, Q$ using an iterative procedure that is described in what follows. Note that a section of a minimum cost q-chain formed by nonserviced edges corresponds, necessarily, to a shortest path in G.

PROCEDURE q-chain

```
Init

  $u(j,q) = \infty \quad \forall j \in V_R, \forall q < 0$
  $u(j,0) = s_{1j} \quad \forall j \in V_R$

Iteration

  For each $q = 1, 2, \ldots, Q$ do:
  Step 1. For each vertex $j \in V'_R$ compute:
    $u(j,q) = \text{Min}\{u(i,q - q_{ij}) + c'_{ij}, \quad \forall (i,j) \in E_R)\}$.
  Step 2. For each vertex $j \in V_R$ compute:
    $u(j,q) = \text{Min}\{u(j,q), \text{Min}\{u(i,q) + s_{ij}; \quad \forall i \neq j, i \in V'_R\}\}$.
```

Note that after Step 1 has been performed, $u(j,q)$ represents the minimum-cost q-chain from the depot to vertex j with load q, with the restriction that the last edge in the q-chain is serviced. In this computation, only costs $u(i,q')$, for $q' < q$, are needed, and, by induction, they have already been computed. In Step 2, it is allowed to traverse (and not service) the last part of the chain (which corresponds to a shortest path). At the end of the procedure, costs $u(j,q)$ will correspond to the above definition. Note that it is not needed to know in advance the true value of $u(i,q)$ in Step 2 because the minimum q-chain corresponding to $u(j,q)$ can be divided into two sections (possibly empty), one (starting from the depot) whose last edge is serviced and the other being a shortest path traveled without servicing any edge. For each q, in Step 1, the required number of operations is of order $2|E_R|$ and, in Step 2, of order $|V_R|^2$. Therefore, the total complexity of the procedure is of order $O(Q(|V_R|^2 + |E_R|))$.

Using costs $u(j,q)$, it can be computed:

$$u_{eq} = \text{Min}\{u(i,q - q_e) + s_{j1}, u(j, q - q_e) + s_{i1}\} + c'_{ij} \qquad \forall e = (i,j) \in E_R, \quad \forall q \in \{0, \ldots, Q\}.$$

Obviously, u_{eq} is a lower bound to the cost of a route that starting from the depot has a total load of q, being $e = (i,j)$, the last serviced edge before returning to the depot. The corresponding "routes," which will be called q-routes, can be obtained as it is usual in the dynamic programming algorithms.

4.2 Bound LB4

Using costs u_{eq}, it is possible to compute a lower bound to the optimal value of the CARP by using a dynamic programming algorithm, similar to the one proposed by Christofides et al. [9] for Node Routing Problems. The procedure can be shortly described as follows:

Denote by R the set of q-routes for all $e = (i,j) \in E_R$ and for all $q \in \{0, \ldots, Q\}$. Required edges are arbitrarily numbered from 1 to $|E_R|$; let $e(i)$ be the required edge in the ith position. Consider the set R as partitioned into $|E_R|$ blocks according to the last required edge serviced in the q-route.

Let $h_i(k,q)$ be the cost of the optimal solution to the problem of selecting, at minimum cost, k q-routes ($k \leq K$), at most one of each block, such that the total load is q and only q-routes in the first i blocks are considered. It is quite obvious that the following recursive equations hold:

$$h_i(k,q) = \text{Min}\{h_{i-1}(k,q), \text{Min}_{q' \in W}\{h_{i-1}(k-1, q-q') + u_{e(i)q'}\}\}$$

for $k = 2, \ldots, K$, $i = k, \ldots, |E_R| - K + k$, $Q_T - (K-k)Q \leq q \leq \text{Min}\{kQ, Q_T\}$, where $W = \{0, 1, 2, \ldots, Q\}$.

The initialization is given by

$$h_i(1,q) = \begin{cases} \text{Min}_{1 \leq j \leq i}\{u_{e(j)q}\} & \text{if } q \in W \\ \infty & \text{otherwise} \end{cases}$$
$$h_1(2,q) = \infty \text{ for all } q.$$

From the above definition, it is clear that a lower bound for the CARP can be obtained by computing

$$h_{|E_R|}(K, Q_T)$$

using the above recursive equations. We can obtain also the corresponding set of K q-routes satisfying the capacity constraints and with total load Q_T. However, some edges can be serviced, either by the same q-route or jointly by the set of q-routes, more than once, while other edges may remain unserviced. The following sections describe the procedures applied in order to improve the lower bound. First, Lagrangean relaxation is applied and, on the other hand, lower bounds u_{eq} are improved.

4.2.1. Lagrangean Relaxation

As was mentioned before, given a feasible solution to the CARP, the corresponding augmented graph has a set of paths, formed by artificial edges, such that every odd-degree vertex of G_R is a terminal vertex of at least one of these paths and at least J paths have the depot as terminal vertex. These constraints, violated by the set of q-routes produced by bound LB4, can be relaxed, in a Lagrangean way, using multipliers π_i, $i \in S \cup \{1\}$ that modify the costs of the shortest paths s_{ij}. The role of these multipliers is similar to that of the dual variables in a minimum-cost matching problem and their initial values are heuristically computed.

More important are the constraints requiring each edge to be serviced exactly once. These constraints can also be incorporated, in a Lagrangean way, using multipliers λ_e, $e \in E_R$, that modify the costs c'_e. Initially, we set $\lambda_e = c'_e$ $\forall e \in E_R$. Then, the subgradient method is used to improve the values of multipliers λ_e and, consequently, the lower bound. Multipliers π_i are not modified during the whole process, as the computational experience showed that the Subgradient Method applied to these multipliers produced no significant improvement in the value of the lower bound.

4.2.2. q-Chains without 2-Loops

The q-chains obtained by the procedure described in Section 3.1 may service edge $e = (i,j)$, while traversing it from i to j and, immediately after, return from j to i and service once more this edge. It is possible to prevent this behavior, thus augmenting the value of the lower bound, without an excessive increment of the computational cost. The resulting q-chains will be referred to as q-chains without 2-loops. The procedure is similar to that of Christofides et al [9, 10] and Kolen et al. [13] for Node Routing Problems.

To avoid the use of a new notation, $u(j,q)$ will denote, in what follows, the cost of the minimum cost q-chain without 2-loops, from the depot to vertex j and with a total load of q.

We will denote by $u(j,q,/i)$ the cost of the minimum-cost q-chain without 2-loops from 1 to j, with a total load of q and such that, if the last edge is serviced, this is not edge (i,j). On the other hand, let $p(j,q)$ be the vertex before vertex j in the q-chain corresponding to $u(j,q)$, with a minus sign if the final section of the q-chain corresponds to a shortest path, which is traversed but not serviced.

Note that $u(j,q,/i) = u(j,q)$, $\forall i \neq p(j,q)$, and, only when $p(j,q) > 0$, the computation of $u(j,q,/i)$ is needed for $i = p(j,q)$.

Therefore, the new procedure is as follows:

PROCEDURE *q*-Chains without 2-Loops

Init

$$u(j,q) = \infty, \quad p(j,q) = 0 \quad \forall j \in V_R, \quad \forall q < 0$$

$$u(j,0) = s_{1j}, \quad p(j,0) = -1 \quad \forall j \in V_R$$

Iteration

For each $q = 1, 2, \ldots, Q$, do:

```
Step 1
For each $j \in V'_R$ do:
(a) $u(j,q) = \text{Min}\{A + c'_{ij},\ (i,j) \in E_R\}$
  where:
    $A = u(i,q - q_{ij}, /j)$    if $j = p(i,q - q_{ij})$ and
    $A = u(i,q - q_{ij})$        otherwise
  Let $i^*$ be the vertex for which the minimum is attained. Do $p(j,q) = i^* > 0$.
(b) $u(j,q,/i^*) = \text{Min}\{u(i,q - q_{ij},/j) + c'_{ij};\ (i,j) \in E_R,\ i \neq i^*\}$

Step 2
For each $j \in V_R$ do:
  $u(j,q) = \text{Min}\{u(j,q), \text{Min}\{u(i,q) + s_{ij};\ \forall i \neq j\}\}$
  If $u(j,q)$ has changed and the minimum is attained for the vertex $i^*$, do
  $p(j,q) = -i^* < 0$.
```

It is not hard to see that, in order to compute costs $u(j,q)$ using this algorithm, the number of operations needed is of the same order than that of the algorithm described in Section 3.1, though, obviously, the computational cost will be greater in practice.

Using the costs $u(j,q)$, the new lower bounds u_{eq} are computed as follows:

$$u_{eq} = \text{Min}\{u(i,q - q_e,/j) + s_{j1},\ u(j,q - q_e,/i) + s_{i1}\} \qquad \forall e = (i,j) \in E_R, \quad \forall q = 0, 1, \ldots, Q.$$

LB4 will denote the lower bound to the CARP that results from applying the mentioned algorithm of Christofides et al. [9], using these improved values u_{eq} in combination with the Subgradient Method explained above.

5. COMPUTATIONAL RESULTS

The lower bounds presented in this paper have been implemented and applied to a set of CARP instances defined on randomly generated graphs whose main characteristics are shown in Table I, where n and m represent, respectively, the number of vertices and the number of edges. Associated to every edge there is a load, a servicing cost, and a traversing cost, generated in the range [1, 15]. In these examples, all the edges are required, i.e., they all have positive loads.

TABLE I. Characteristics of the graphs.

	1	2	3	4	5	6	7	8	9	10
n	24	24	24	41	34	31	40	30	50	50
m	39	34	35	69	65	50	66	63	92	97
Q_T	358	310	137	627	614	457	562	566	654	704
C_T	220	256	89	465	510	297	352	483	405	585

TABLE II. Computational results.

Inst. I	K_0	Q	UB	PEARN	ZAW1	ZAW2	LB1	LB2	LB3	LB4	%
1.A	2	200	256	$\underline{247}$	230	$\underline{247}$	$\underline{247}$	$\underline{247}$	$\underline{247}$	244	3.5
1.B	3	120	260	$\underline{247}$	230	$\underline{247}$	$\underline{247}$	$\underline{247}$	$\underline{247}$	244	5.0
2.A	2	180	324	296	281	296	296	296	$\underline{297}$	296	8.3
2.B	3	120	346	305	309	$\underline{318}$	305	$\underline{318}$	309	308	8.1
3.A	2	80	108	$\underline{103}$	94	$\underline{103}$	$\underline{103}$	$\underline{103}$	$\underline{103}$	$\underline{103}$	4.6
3.B	3	50	115	105	100	$\underline{108}$	105	$\underline{108}$	107	105	6.1
4.A	3	225	536	514	478	514	514	514	$\underline{516}$	513	3.7
4.B	4	170	576	517	484	518	518	518	$\underline{522}$	515	9.4
4.C	5	130	624	521	490	523	524	524	$\underline{528}$	517	15.4
5.A	3	220	594	$\underline{562}$	518	$\underline{562}$	$\underline{562}$	$\underline{562}$	$\underline{562}$	547	5.4
5.B	4	165	620	566	532	567	566	567	$\underline{580}$	551	6.4
5.C	5	130	646	582	546	581	582	582	$\underline{598}$	555	7.4
6.A	3	170	337	$\underline{330}$	313	$\underline{330}$	$\underline{330}$	$\underline{330}$	$\underline{330}$	325	2.1
6.B	4	120	351	334	317	334	334	334	$\underline{336}$	329	4.3
7.A	3	200	382	$\underline{382}$	365	$\underline{382}$	$\underline{382}$	$\underline{382}$	$\underline{382}$	376	0.0
7.B	4	150	414	$\underline{382}$	365	$\underline{382}$	$\underline{382}$	$\underline{382}$	$\underline{382}$	376	7.7
8.A	3	200	556	$\underline{522}$	491	$\underline{522}$	$\underline{522}$	$\underline{522}$	$\underline{522}$	519	6.1
8.B	4	150	566	528	497	528	528	528	$\underline{531}$	525	6.2
9.A	3	235	462	$\underline{450}$	412	$\underline{450}$	$\underline{450}$	$\underline{450}$	$\underline{450}$	444	2.6
9.B	4	175	484	$\underline{453}$	418	$\underline{453}$	$\underline{453}$	$\underline{453}$	$\underline{453}$	447	6.4
9.C	5	140	494	456	424	$\underline{459}$	$\underline{459}$	$\underline{459}$	$\underline{459}$	453	7.1
10.A	3	250	658	$\underline{637}$	597	$\underline{637}$	$\underline{637}$	$\underline{637}$	$\underline{637}$	632	3.2
10.B	4	190	665	641	601	641	641	641	$\underline{645}$	636	3.0
10.C	5	150	684	647	607	647	649	649	$\underline{653}$	640	4.5

For each graph I, several instances have been generated (denoted by I.A, I.B, . . .) by varying the capacity of the vehicles (and, therefore, K_0). The computational results of bounds LB1, LB2, LB3, and LB4 are presented in Table II, which contains as well: UB, an upper bound for the CARP using K_0 vehicles, obtained by applying the algorithm described in Benavent et al. [4]; the lower bound proposed by Pearn [16]; those proposed by Zaw Win [19]: ZAW1 and ZAW2, described in Section 2; and the deviation percentage of the best lower bound (underlined) with respect to the upper bound. In the computation of lower bounds LB3 and LB4, the number of vehicles was set equal to K_0 and 50 iterations of the Subgradient Method were performed to compute LB4. Table III compares the performance of the lower bounds for the instances in Table II. Except for bound LB4, all the computational results were obtained with an IBM PS/2 80-071. Because of memory requirements, this last bound

TABLE III. Comparison of performance of lower bounds for instances in Table II.

	PEARN	ZAW1	ZAW2	LB1	LB2	LB3	LB4
Average % under UB	6.4	12.0	6.1	6.3	6.1	5.8	7.4
Maximum % under UB	16.5	21.5	16.2	16.0	16.0	15.4	17.1
No. times bound is best or tied for best	11	0	14	12	14	22	1

TABLE IV. CPU time in seconds on an IBM PS/2 80-071.

	PEARN	ZAW1	ZAW2	LB1	LB2	LB3	LB4
Average CPU time	8.16	0.12	2.11	1.21	2.28	14.55	19.28[a]
Maximum CPU time	25.86	0.21	5.43	2.80	5.49	34.89	44.07[a]

[a] CPU time in seconds on an IBM 3090.

was run in a mainframe, an IBM 3090. The maximum and the average CPU times in seconds for each lower bound on these instances are shown in Table IV.

From the results included in Tables II–IV, several comments can be made: First, it becomes clear that ZAW1 is the fastest among all procedures but it produces the worst bounds in this set of instances. As was shown theoretically, LB1 dominates the Pearn bound and it produced a bound strictly greater than the Pearn bound in four instances; furthermore, LB1 is almost seven times faster on average than is the Pearn bound. LB1 and ZAW2 produce quite similar results on this set of instances, but LB1 is 1.75 times faster on average than ZAW2. As was shown, bound LB2 outperforms LB1 and ZAW2 and it is only slightly slower than ZAW2. Bound LB3 is better than LB2 but it is much more time-consuming, almost seven times on average, and it is only a valid lower bound when the number of vehicles is fixed and equal to K_0. Finally, bound LB4, in contrast to what we expected, produces rather bad results and it requires much more computational effort. Note, however, that bound LB4, if used in a branch-and-bound method to find an optimal solution to the CARP, can provide q-routes that may become feasible solutions to the problem after a certain number of branches. The other bounds seem to be less appropriate, at least a priori, to be implemented in exact algorithms, because they only produce a graph (that includes a set of artificial edges) in which, deciding whether it exists, a feasible solution using only these artificial edges is itself an NP-complete problem.

As most lower bound procedures tend to perform well and in a similar way on the set of instances of Table II, we have generated, for further testing, 10

TABLE V. Computational results for special instances.

Inst. I	K_0	Q	UB	PEARN	ZAW1	ZAW2	LB1	LB2	LB3	LB4	%
1.C	8	45	370	267	268	280	279	280	285	274	22.9
2.C	8	40	606	348	479	482	386	482	432	357	20.4
3.C	7	20	174	116	134	139	123	140	131	113	19.5
4.D	9	75	757	537	526	547	558	565	570	549	24.7
5.D	9	75	839	626	600	628	656	656	672	571	19.9
6.C	10	50	448	350	367	368	372	372	372	362	16.9
7.C	9	65	480	396	393	403	402	403	402	398	16.0
8.C	9	65	742	553	557	574	587	587	603	572	18.7
9.D	10	70	597	475	460	492	493	493	497	483	16.7
10.D	10	75	822	674	641	675	697	697	697	674	15.2

TABLE VI. Comparison of performance of lower bounds for instances in Table V.

	PEARN	ZAW1	ZAW2	LB1	LB2	LB3	LB4
Average % under UB	26.1	23.6	20.9	22.5	19.7	20.5	25.8
Maximum % under UB	42.5	30.5	27.7	36.3	25.3	28.7	41.0
No. times bound is best or tied for best	0	0	2	2	5	7	0

additional instances by augmenting the number of vehicles and reducing their capacities, thus strongly decreasing the number of edges that a vehicle will probably service.

Tables V and VI, which parallel Tables II and III, respectively, show the computational results of the various lower bounds for this new set of instances. The computation times do not change significantly in these instances, so they are not shown. As we suspected, all the lower bounds get worse, with an average percentage deviation from the upper bound that ranges from 19.7 for LB2 to 26.1 for PEARN. The Pearn bound becomes the worst in this set of instances, since it tends to perform worse, when compared with LB1, as the number of vehicles increases (see the proof of Theorem 2). On the other hand, ZAW1 is the lower bound with the smallest worsening when applied to this new set of instances. This result can be explained by the fact that this is the only lower bound, among those studied here, that does not take into account the parity of the vertices of the graph. Since this factor remains unchanged when the capacity of the vehicles decreases and the required computational effort is minimum, ZAW1 is a valuable option when dealing with large instances in which a big number of vehicles is needed. Finally, considering all the tested instances, bound LB2 has the best performance, taking into account both computation times and quality of the results, which is not surprising because it synthesizes the best features from most of the other lower-bounding procedures.

We are very grateful for the comments and suggestions of two unknown referees.

REFERENCES

[1] A. Assad, W. Pearn, and B. Golden. The capacitated postman problem: Lower bounds and solvable cases. *Am. J. Math. Management Sci.* **7** (1,2) (1987) 63–88.

[2] E. Beltrami and L. Bodin, Networks and vehicle routing for municipal waste collection. *Networks* **4** (1) (1974) 65–94.

[3] E. Benavent, V. Campos, A. Corberán, and E. Mota, Problemas de Rutas por Arcos. *Questiió* **7**(3) (1983) 479–490.

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

[5] L. Bodin and B. Golden, Classification in vehicle routing and scheduling. *Networks* **11**(2) (1981) 97–108.

[6] L. Bodin, B. Golden, A. Assad, and M. Ball, Routing and scheduling of vehicles and crews. *Comput. Operations Res.* **10** (1983) 63–211.

[7] L. Chapleau, J. Ferland, G. Lapalme, and J. M. Rousseau, A parallel insert method for the capacitated arc routing problem. *Operations Res. Lett.* 3 (1984) 95–99.

[8] N. Christofides, The optimum traversal of a graph. *Omega* 1(6) (1973) 719–732.

[9] N. Christofides, P. Mingozzi, and P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Math. Programming* **20** (1981) 255–282.

[10] N. Christofides, P. Mingozzi, and P. Toth, State-space relaxation procedures for the computation of bounds to routing problems. *Networks* **11**(2), (1981) 145–164.

[11] B. Golden and R. Wong, Capacitated arc routing problems. *Networks* **11**(2) (1981) 305–315.

[12] B. Golden, J. S. De Armon, and E. K. Baker, Computational experiments with algorithms for a class of routing problems. *Comput. Operations Res.* **10**(1) (1983) 47–59.

[13] A. Kolen, A. H. G. Rinnooy-Kan, and H. Trienekens, Vehicle routing with time windows. Report 8433/O, Econometric Institute, Erasmus University, Rotterdam (1984).

[14] J. K. Lenstra and A. H. G. Rinnooy-Kan, Complexity of vehicle routing and scheduling problems. *Networks* **11**(2) (1981) 221–227.

[15] J. W. Male and J. C. Liebman, Districting and routing for solid waste collection. *J. Environmental Eng. Div.* **104** (1978) EE1.

[16] W. L. Pearn, New lower bounds for the capacitated arc routing problems. *Networks* **18** (1988) 181–191.

[17] Y. Saruwatari, R. Hirabayashi, and N. Nishida, Node duplication lower bounds for the capacitated arc routing problem. *Proceedings of the Third Euro Club for Combinatorial Optimization,* Barcelona (1990).

[18] H. Stern and M. Dror, Routing electric meter readers. *Comput. Operations Res.* 6 (1979) 209–223.

[19] Zaw Win: Contributions to routing problems. PhD Dissertation, Universität Augsburg (1988).

Received April 1989
Accepted May 1992