Journal of Mathematical Modelling and Algorithms **3:** 209–223, 2004.

Tabu Search Heuristics for the Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions

GIANPAOLO GHIANI[1], FRANCESCA GUERRIERO[2], GILBERT LAPORTE[3] and ROBERTO MUSMANNO[2]
[1]*Dipartimento di Ingegneria dell'Innovazione, Università di Lecce, Strada per Arnesano, 73100 Lecce, Italy*
[2]*Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy*
[3]*Canada Research Chair in Distribution Management, HEC Montréal, 3000, Chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7*

Abstract. This paper deals with the *Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions* (CLARPIF), a variant of the classical *Capacitated Arc Routing Problem* (CARP), in which vehicles may unload or replenish at intermediate facilities and the length of any route may not exceed a specified upper bound. Three heuristics are developed for the CLARPIF: the first is a constructive procedure based on a partitioning approach while the second and the third are tailored Tabu Search procedures. Computational results on a set of benchmark instances with up to 50 vertices and 92 required edges are presented and analyzed.

Mathematics Subject Classifications (2000): 90B60, 9B10, 68T20.

Key words: capacitated arc routing problem, intermediate facilities, capacity and distance restrictions.

1. Introduction

The aim of this paper is to describe three heuristics for the *Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions* (CLARPIF), a variant of the classical *Capacitated Arc Routing Problem* (CARP) [15]. Both these problems, in their undirected versions, are defined on a graph $G = (V, E)$ where $V = \{v_1, \dots, v_n\}$ is a vertex set and E is a set of edges (v_i, v_j) $(i < j)$, including a subset R of *required* edges. Each edge of R must be serviced once but can be traversed several times. Let V_R be the set of vertices v_i such that an edge (v_i, v_j) exists in R. With each edge $e = (v_i, v_j)$ are associated a non-negative traversal *cost* or *length* $c_e = c_{ij}$ and a non-negative *demand* $q_e = q_{ij}$. If $e \notin R$, then $q_e = 0$. Denote by d_{ij} the length of a shortest chain between two vertices v_i and v_j. Vertex v_1 denotes a *depot* at which a fleet of identical *vehicles* is based. In our version of the problem, vehicles bear no fixed costs and their number is a decision variable.

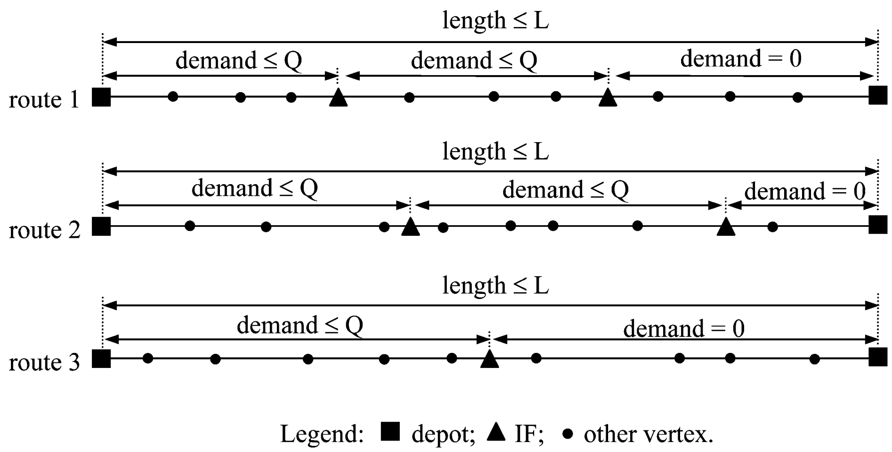

Figure 1. Feasible CLARPIF solution (routes are traversed from left to right).

The set V contains a subset I of *intermediate facilities* (IFs), possibly including v_1. The CARP (see, e.g., [1], or [8, 9], or [11] for surveys) consists of designing a least cost set of routes such that:

(a) every route starts and ends at the depot;
(b) the total demand serviced by any route may not exceed a given vehicle capacity Q.

The CARP reduces to the *Rural Postman Problem* (RPP) whenever $\sum_{e \in R} q_e \leqslant Q$. The CLARPIF is to design a least cost set of *vehicle routes* in such a way that (see Figure 1):

(c) every route starts and ends at the depot;
(d) every required edge is serviced by exactly one vehicle;
(e) in every route the total demand collected between the depot and the first IF, or between two successive IFs, may not exceed the vehicle capacity;
(f) in every route, if the last IF is not the depot, then the final chain between that facility and the depot may not have any positive demand;
(g) the length of any route may not exceed a preset bound L.

A different but conceptually equivalent version of the CLARPIF is where the vehicle makes deliveries instead of collections. Then it must replenish at IFs in order to satisfy the demand until it reaches the next facility or the depot. Applications of the first version of the CLARPIF arise in garbage collection where the vehicles make intermediate visits at dump sites or incinerators. Such systems are described in [22] for the city of Calgary in Canada, and in [13] for the town of Castrovillari in Southern Italy. Instances of the second version are encountered in road gritting where IFs are conveniently located sand or chemical boxes (see, e.g., [20, 7]). What follows applies to the first version of the CLARPIF but adaptations of our results to the second version are straightforward.

The CLARPIF is NP-hard since it reduces to the RPP, shown to be NP-hard by Lenstra and Rinnooy Kan [19], whenever $I = \{v_1\}$, $Q = \sum_{e \in R} q_e$ and $L = \infty$. If $L = \infty$, then the CLARPIF becomes the *Capacitated Arc Routing Problem with Intermediate Facilities* (CARPIF) introduced by Ghiani, Improta and Laporte [12].

The remainder of paper is organised as follows. In Section 2 we investigate some properties of the problem. A constructive heuristic is described in Section 3, while tabu search procedures are presented in Section 4. Computational results are provided in Section 5, followed by the conclusion in Section 6.

2. Problem Properties

Unlike the CARP and the CARPIF, the CLARPIF can be infeasible. The feasibility of a CLARPIF instance can be checked in polynomial time.

PROPOSITION 1. *A CLARPIF instance is feasible if and only if* $c_e + \min(L'_e, L''_e) \leqslant L$, $e = (v_i, v_j) \in R$, *where* $L'_e = d_{1i} + \min_{v_k \in I}[d_{jk} + d_{1k}]$ *and* $L''_e = d_{1j} + \min_{v_k \in I}[d_{ik} + d_{1k}]$.

Proof. The condition is sufficient. The length of a route r_e servicing a single edge $e = (v_i, v_j)$ is equal to $c_e + L'_e$ or $c_e + L''_e$, depending on whether edge e is serviced from v_i to v_j or from v_j to v_i. Consequently, if the condition holds, each required edge can be feasibly serviced.

The condition is necessary. Any route servicing a required edge $e = (v_i, v_j)$ has a length greater than or equal to $c_e + \min(L'_e, L''_e)$. Hence, if a feasible solution exists, then the condition holds. □

Once the feasibility check described in Proposition 1 has been performed successfully, the following reduction rule can sometimes be used to remove dominated IFs.

REDUCTION RULE. For each $v_i, v_j \in V_R$, let $f^*_{ij} = \arg\min_{f \in I}\{d_{if} + d_{fj}\}$ and, for each $v_i \in V_R$, let $f^{**}_i = \arg\min_{f \in I}\{d_{if} + d_{1f}\}$, and reset

$$I := \left(\bigcup_{v_i, v_j \in V_R} \{f^*_{ij}\} \right) \cup \left(\bigcup_{v_i \in V_R} \{f^{**}_i\} \right).$$

3. A Constructive Heuristic

The constructive heuristic we have developed computes a CLARPIF feasible solution by optimally partitioning an RPP tour into a set of feasible routes, in the spirit of Beasley's [2] route-first/cluster-second heuristic for the capacitated *Vehicle Routing Problem*, of Jansen's [18] procedure for the CARP and of Ghiani, Improta and Laporte's [12] algorithm for the CARPIF. More formally, the heuristic can be described as follows.

Step 1. Determine an RPP feasible tour by means of Frederickson's heuristic [10], which first constructs a shortest tree spanning all components of required edges, and then computes an optimal matching of odd degree vertices.

Step 2. For each orientation of the RPP solution, let $e_r = (v_{i_r}, v_{j_r})$ be the rth serviced edge of the solution starting from the depot. For every $h, k \in \{1, \ldots, |R|\}$, $h \leqslant k$, determine the least cost route servicing required edges $e_h, \ldots, e_k$ by computing a shortest path on an auxiliary directed graph G'. Let r_{hk} be the best of the two routes corresponding to the two orientations of the RPP solution.

Step 3. Discard every route $r_{hk}, h, k \in \{1, \ldots, |R|\}$, $h \leqslant k$, whose length exceeds L.

Step 4. Determine a least cost set of feasible routes covering all required edges by computing a shortest path on an auxiliary directed graph G''.

It is worth noting that, if the CLARPIF is feasible, this algorithm always finds a feasible solution. In fact, if the hypotheses of Proposition 1 hold, the solution $\{r_{11}, \ldots, r_{|R||R|}\}$ may be generated by the procedure. In what follows, a more detailed description of Steps 2 and 4 is given.

3.1. DETAILED DESCRIPTION OF STEP 2

Construct the auxiliary directed graph $G' = (U, A_1 \cup A_2 \cup A_3)$ as follows. The vertex subset $U = \{a_1, \ldots, a_{|R|}\} \cup \{b_1, \ldots, b_{|R|}\} \cup W$ is made up of $|R|$ sources $a_1, \ldots, a_{|R|}$, $|R|$ sinks $b_1, \ldots, b_{|R|}$, and a vertex $w_{hk} \in W$ for each sequence $e_h, \ldots, e_k$ ($h, k \in \{1, \ldots, |R|\}$, $h \leqslant k$), such that $\sum_{r=h}^{k} q_{i_r j_r} \leqslant Q$. Let λ_{hk} be the length of the chain of the RPP solution corresponding to the sequence $e_h, \ldots, e_k$. The arc subset A_1 contains an arc (a_h, w_{hk}) of cost d_{1i_h} from every source a_h to every vertex w_{hk}. The arc subset A_2 contains an arc (w_{hk}, b_k) of cost $[\lambda_{hk} + \min_{f \in I}(d_{j_k f} + d_{f1})]$ from every vertex w_{hk} to every sink b_k. The arc subset A_3 contains an arc (w_{hk}, w_{k+1p}) of cost $[\lambda_{hk} + \min_{f \in I}(d_{j_k f} + d_{f i_{k+1}})]$ from every vertex w_{hk} to every vertex w_{k+1p}. See Figure 2 for a sample auxiliary graph G', where

$$\sum_{r=1}^{2} q_{i_r j_r} \leqslant Q, \qquad \sum_{r=1}^{3} q_{i_r j_r} > Q, \qquad \sum_{r=2}^{3} q_{i_r j_r} \leqslant Q,$$
$$\sum_{r=2}^{4} q_{i_r j_r} > Q, \qquad \sum_{r=3}^{|R|} q_{i_r j_r} \leqslant Q.$$

The least cost route servicing required edges $e_h, \ldots, e_k$ is then obtained by computing a shortest path from the source a_h to the sink b_k on G'.

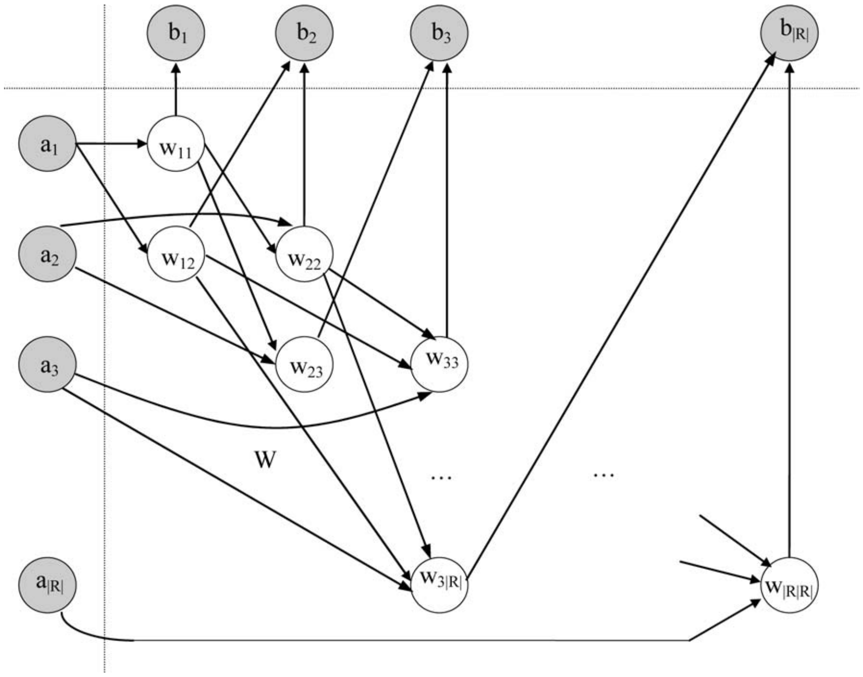

Figure 2. Computing feasible routes on G'.

3.2. DETAILED DESCRIPTION OF STEP 4

Construct the auxiliary directed graph $G'' = (S, A_4 \cup A_5 \cup A_6)$ as follows. The vertex subset $S = (\{s_1, s_2\} \cup T)$ is made up of a source s, a sink t, and a vertex $t_{hk} \in T$ for each feasible route r_{hk}. Let δ_{hk} $(\leqslant L)$ be the length of route r_{hk}. The arc subset A_4 contains a zero cost arc (s_1, t_{1k}) from source s_1 to every vertex t_{1k}. The arc subset A_5 contains an arc $(t_{h|R|}, s_2)$ of cost $\delta_{h|R|}$ from every vertex $t_{h|R|}$ to the sink s_2. The arc subset A_6 contains an arc (t_{hk}, t_{k+1p}) of cost δ_{hk} from every vertex t_{hk} to every vertex t_{k+1p}. See Figure 3 for a sample auxiliary graph G'', where routes r_{1k} $(k \in \{4, \dots, |R|\})$ are infeasible. A least cost set of routes covering all required edges is then obtained by computing a shortest path from s_1 to s_2 on G''.

4. Tabu Search Heuristics

This section contains a description of two tabu search heuristics for the CLARPIF (referred to as TS1 and TS2 in the following). Tabu search is a well-known local search heuristic that moves at each iteration from a solution to its best neighbor, even if this causes a deterioration of the objective function [14]. To avoid cycling, solutions possessing some attributes of recently explored solutions are declared forbidden or tabu for a number of iterations. Intensification and diversification strategies are usually employed to improve the search procedure. In the first case, the search is accentuated in promising regions of the feasible domain. In the second

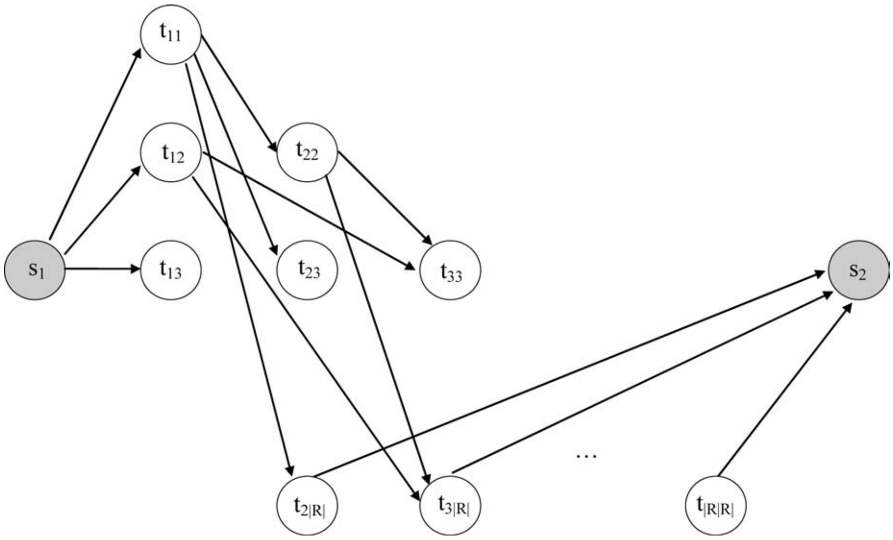

Figure 3. Computing a least cost set of routes on G''.

case, an attempt is made to consider solutions in a broader area of the search space. To produce good results, any implementation of the TS must be crafted to suit the structure of the problem at hand. In the following, we outline the main ingredients of the procedure.

We first describe the features common to TS1 and TS2 procedures. A solution is a set S of m routes. Denote by z_{ij}^r the number of times edge $e = (v_i, v_j)$ is traversed in route R_r. In every route a chain between the depot and the first IF, or between two successive IFs, is called a *route segment*. Each route R_r is a sequence of n_r route segments and a final shortest path between the last IF and the depot. Let y_{ij}^{rh} be a binary constant equal to 1 if edge $e = (v_i, v_j) \in R$ is serviced by route segment h of the route r, and 0 otherwise. In both TS1 and TS2 we allow infeasible intermediate solutions, with an appropriate penalty in the objective function as in Gendreau, Hertz and Laporte [16], and Hertz, Laporte and Mittaz [17]. More precisely, with any feasible solution S, we associate the objective function

$$F_1(S) = \sum_r \sum_{(v_i, v_j) \in E} c_{ij} z_{ij}^r.$$

Also, with any solution S (feasible or not), we associate the objective:

$$F_2(S) = F_1(S) + \alpha \sum_r \sum_{h=1}^{n_r} \left[\left(\sum_{(v_i, v_j) \in R} q_{ij} y_{ij}^{rh} \right) - Q \right]^+ + \\ + \beta \sum_r \left[\left(\sum_{(v_i, v_j) \in E} c_{ij} z_{ij}^r \right) - L \right]^+,$$

where $[x]^+ = \max(0, x)$, and α and β are positive penalty parameters. If S is feasible, then $F_1(S)$ and $F_2(S)$ coincide; otherwise $F_2(S)$ embodies two penalty terms for excess vehicle capacity and excess route duration. Parameters α and β are adjusted dynamically as in Gendreau, Hertz and Laporte [16] and Hertz, Laporte and Mittaz [17]. If the last η iterations were all feasible with respect to the capacity restriction, then $\alpha := \alpha/2$. If they were all infeasible, then $\alpha := 2\alpha$; otherwise α is unchanged. Similarly, if the last η iterations were all feasible with respect to the distance restriction, then $\beta := \beta/2$. If they were all infeasible, then $\beta := 2\beta$; otherwise β is unchanged. This strategy produces a mix of feasible and infeasible solutions and removes the need to calibrate α and β a priori. At any step of either TS1 or TS2, F_1^* and F_2^* represent the lowest value of $F_1(S)$ and $F_2(S)$ obtained during the search process, respectively. In addition, S^* indicates the best known feasible solution.

We also use a variable tabu list length in the spirit of Taillard [21]. A chosen move is declared tabu for a number of iterations randomly chosen in a range $[\theta_1, \theta_2]$. After a number of preliminary tests, we set $\eta = 10$, $\theta_1 = 5$ and $\theta_2 = 10$.

4.1. PROCEDURE TS1

TS1 is based on a main subroutine, named SEARCH, which attempts to improve upon an initial solution S_s, using tabu search. The neighbours of a solution are obtained by transferring a serviced edge into a different route segment (*edge transfer move*), by inserting an IF in a route segment (*IF insertion move*), by ejecting an IF from a route (*IF ejection move*), by splitting a route (*route split move*), or by merging two routes (*route merge move*).

Edge Transfer Move. A serviced edge $e = (v_i, v_j) \in R$ is removed from its current route segment and inserted into another route segment. For each service direction of edge e, two cases may occur: (a) the edge is inserted between two serviced edges (v_h, v_k) and (v_r, v_s); (b) the edge is inserted between an IF v_k and a serviced edge (v_r, v_s), or between a serviced edge (v_s, v_r) and an IF v_k. If edge e is serviced from v_i to v_j, in both cases the shortest chain (without serviced edges) between v_k and v_r is replaced by the shortest chain (without serviced edges) between v_k and v_i, followed by (v_i, v_j) and by the shortest chain (without serviced edges) between v_j and v_r. Similarly, if edge e is serviced from v_j to v_i, the shortest chain (without serviced edges) between v_k and v_r is replaced by the shortest chain (without serviced edges) between v_k and v_j, followed by (v_j, v_i) and by the shortest chain (without serviced edges) between v_i and v_r. In order to speed up the procedure, a small subset of candidate edges is selected as follows:

- for each vehicle route that violates the distance restriction, the serviced edge whose removal would make the route length closer to L (and no greater than L, if possible) is selected;

- for each route segment servicing a demand greater than Q, the serviced edge whose removal would make the segment's demand closer to Q (and no greater than Q, if possible) is selected;
- for each route segment, the serviced edge whose removal would decrease the segment length the most is selected.

Each selected edge $e = (v_i, v_j)$ may be inserted in a route segment between an IF (or the depot) and the first serviced edge, between two adjacent serviced edges, or between the last serviced edge and an IF. We require that the length of each shortest path linking v_i and v_j to the route segment be less than or equal to the average distance between two vertices in V_R.

IF Insertion Move. A copy of an IF is inserted in a route segment. In order to speed up the algorithm, this move is performed uniquely on route segments whose total demand exceeds Q.

IF Ejection Move. A copy of an IF is removed from a route. This move is performed uniquely on pairs of adjacent route segments whose demand is less than or equal to Q.

Route Split Move. A route is partitioned in two distinct routes. In order to quicken the algorithm, a route can be split uniquely after a dump at an IF.

Route Merge Move. Two vehicle routes are merged.

The search procedure SEARCH is governed by a vector P of parameters

$$P = (q, n_{\text{MAX}})$$

defined as follows:

- q: a fraction of randomly chosen required edges on which the edge transfer move can be applied;
- n_{MAX}: the maximum number of iterations.

A step-by-step description of SEARCH procedure is provided in Figure 4.

Starting from an initial solution, the tabu search procedure performs a diversification and an intensification phase. These two phases are obtained by executing the search procedure on the basis of different values P_1 and P_2 of the parameters. As a rule, in the intensification phase a larger number of iterations should be performed, and, at each step, a larger number of moves should be evaluated. A step-by-step description of TS1 is provided in Figure 5. We have carried out a large number of preliminary tests in order to tune the algorithm. As a result, we set $P_1 = (1/4, |R|)$ and $P_2 = (1, 2|R|)$.

Procedure SEARCH (S_s, P)
Step 0 (*Initialisation*)
Set the iteration count $t := 0$; set $S_0 = S_s$; no move is tabu.
Step 1 (*Move selection*)
Determine the set W of all candidate moves. W includes:

- the edge transfer moves corresponding to a fraction q of the candidate edges;
- every IF insertion move;
- every IF ejection move;
- every route split move;
- every route merge move.

Step 2 (*Move evaluation*)
Evaluate each move $M \in W$ and determine the corresponding solution S. If M is tabu, S is discarded unless the aspiration criterion is satisfied, i.e. S is feasible and $F_1(S) < F_1^*$, or S is infeasible and $F_2(S) < F_2^*$.
Step 3 (*Best move identification*)
Identify the best move, i.e. the move M_b corresponding to the solution:

$$S_b = \arg\min_{M \in W} F_2(S).$$

Step 4 (*Solution and parameters update*)
Implement M_b; set $S_{t+1} = S_b$; add the reverse of M_b to the tabu list; update F_1^*. F_2^* and S^*; update α and β; set $t = t + 1$.
Step 5 (*Stopping check*)
If $t \leqslant n_{\text{MAX}}$ go to Step 1, otherwise STOP.

Figure 4. Step-by-step description of the SEARCH procedure.

Step 1 (*Initialization*)
Set $\alpha = \beta = 1$;
Step 2 (*Initial solution*)
Determine an initial solution S' made up of $|R|$ routes, each of which services a single required edge. Set $F_1^* = F(S')$; $F_2^* = F_1^*$; $S^* = S'$.
Step 3 (*Diversification phase*)
Call SEARCH (S', P_1).
Step 4 (*Intensification phase*)
Call SEARCH (S^*, P_2).
Step 5 (*Final solution*)
S^* is the best known feasible solution.

Figure 5. Step-by-step description of the TS1 procedure.

4.2. PROCEDURE TS2

Procedure TS2 is based is a modification of the CARPET algorithm proposed by Hertz, Laporte and Mittaz [17] for the CARP. In what follows, the main components of TS2 not yet illustrated are described.

An initial solution is obtained by applying the constructive procedure described in Section 3. The subsequent search is based on seven main subroutines. The first routine, named SHORTEN, is based on the observation that if a tour T contains a chain P of traversed (but not serviced) edges, then T can be eventually shortened by replacing P by a shortest chain linking the endpoints of P. Procedure SWITCH modifies the order in which required edges are serviced on a given tour. Given a covering tour T for a set R of required edges, and a nonrequired edge (v, w), procedure ADD builds a new tour covering $R \cup \{(v, w)\}$. Alternatively, given a required edge (v, w) in R, procedure DROP builds a new tour covering $R \backslash \{(v, w)\}$. The CUT procedure decomposes an RPP solution into a set of feasible routes. In our implementation, CUT is made up of Steps 2–4 of the partitioning procedure described in Section 3, i.e. a two stage exact partitioning scheme is used. We have chosen an exact approach since heuristic partitioning methods (like the one used by Hertz, Laporte and Mittaz [17] for the CARP) perform poorly when applied to the more complex CLARPIF. Given a solution made up of multiple routes, PASTE creates a single route. Finally, POSTOPT procedure attempts to identify a better solution by applying PASTE, SWITCH, CUT and SHORTEN. For more details on these routines, the interested reader is referred to Hertz, Laporte and Mittaz [17].

To define the neighborhood $N(x)$ of a solution x, we consider a route T of x, an edge (v, w) serviced in T, and another route T' containing only the depot (the vehicle servicing T' is idle) or a required edge "close" to v or w. At each step of the search all neighbors of the current solution are generated and their objective function values are computed, yielding the new current solution. Procedures DROP and ADD are then executed for each required edge and for each route, in an attempt to reduce solution cost. Finally, POSTOPT is called every γ iterations. In our implementation we have set $\gamma = 10$. The search ends when a proven optimal solution has been obtained, or when the number of consecutive iterations without improvement in F_1^* and F_2^* reaches a value ρ. In our implementation, we have used $\rho = 100$.

5. Computational Results

The constructive heuristic and the tabu search procedures were coded in C and run on a Pentium-1GHz personal computer. The procedures were tested on two sets of benchmark instances adapted for the CLARPIF. The first set includes the 25 CARP instances of DeArmon [4] from which instances 8 and 9 were removed because they contained inconsistencies. Also, in instance 21, the isolated vertex 11 was removed. The size of these instances ranges from 7 to 27 vertices and from 11 to 55 edges, all of which are required. For each instance a single IF was located at vertex $|V|$ and the maximum route length (column 4 of Table I) was generated in such a way that the number of routes is uniformly distributed between 5 and 20. The second set is made up of 28 CARP instances introduced in Benavent et al. [3]. These instances contain between 24 and 50 vertices, and between 34 to 92 edges, all of which are required. Two IFs were located at vertices $\lfloor |V|/2 \rfloor$ and $2\lfloor |V|/2 \rfloor$. In addition, the maximum route length (column 4 of Table II) was generated in such a way that the number of routes is uniformly distributed between 5 and 20.

On the first set of instances we also run a lower-bounding procedure [5] based on a relaxation of an integer linear formulation of the problem. This procedure applies whenever there is a single intermediate facility. Lower bounding algorithms for the general case (e.g., instances with two or more intermediate facilities) are not currently available. Their design is a research topic in its own.

Computational results are presented in Tables I and II. The column headings are as follows:

$|V|$: number of vertices;

$|E|$: number of edges (all required);

L : maximum route length;

LB : lower bound provided by the De Rosa et al. [5] algorithm;

UB_1 : solution value provided by the constructive heuristic;

SEC_1 : computing time in seconds for UB_1;

UB_2 : solution value provided by the TS1 heuristic;

SEC_2 : computing time in seconds for UB_2;

UB_3 : solution value provided by the TS2 heuristic;

SEC_3 : computing time in seconds for UB_3;

$\mathrm{UB}_3/\mathrm{UB}_1$: UB_3 over UB_1 ratio;

$\mathrm{UB}_3/\mathrm{UB}_2$: UB_3 over UB_2 ratio.

Computational results indicate that UB_3 is always better than UB_1 and UB_2. The average percentage deviation of UB_3 over UB_1 is 18.69 and 21.75 for DeArmon and Benavent instances, respectively. Moreover, the average percentage deviation of UB_3 over UB_2 is 10.62 and 15.23 for DeArmon and Benavent instances, respectively. This improvement comes at the expense of a moderate additional computation time. Indeed, the constructive heuristic usually takes just a few seconds while TS2 requires hundreds of seconds on average. This increase in computation time is reasonable in garbage collection and salt gritting applications. As expected [8, 9], lower bounds are not that helpful to assess the performance of the heuristics. In fact, the average percentage deviation of the best heuristic solution over the lower bound value is 14.22. Both instances and results are available at `http://persone.dii.unile.it/ghiani/`.

Table I. Results for the DeArmon instances

Instance	$\lvert V \rvert$	$\lvert R \rvert$	L	LB	UB_1	SEC_1	UB_2	SEC_2	UB_3	SEC_3	UB_3/UB_1	UB_3/UB_2
1	12	22	64	316	421	1	419	3	345	139	0.82	0.82
2	12	26	63	339	493	1	425	10	353	188	0.72	0.83
3	12	22	63	283	385	1	361	5	311	166	0.81	0.86
4	11	19	70	304	503	1	400	4	361	76	0.72	0.90
5	13	26	88	596	795	1	779	14	658	292	0.83	0.84
6	12	22	87	409	722	1	491	7	429	133	0.59	0.87
7	12	22	65	394	543	1	493	9	399	205	0.73	0.81
10	27	46	52	404	556	1	514	217	512	296	0.92	1.00
11	27	51	48	325	465	1	442	312	370	591	0.80	0.84
12	12	25	63	368	668	1	495	16	461	255	0.69	0.93
13	22	45	66	579	871	1	720	192	694	949	0.80	0.96
14	13	23	114	737	1105	1	1011	9	987	222	0.89	0.98
15	10	28	128	584	688	1	678	21	614	358	0.89	0.91
16	7	21	20	120	192	1	154	5	148	152	0.77	0.96
18	8	28	14	137	179	1	183	15	158	750	0.88	0.86
20	9	36	23	216	290	1	280	44	250	446	0.86	0.89
22	11	22	28	165	203	1	215	5	181	153	0.89	0.84
23	11	33	17	164	220	1	207	42	198	486	0.90	0.96
24	11	44	16	236	295	1	276	153	242	748	0.82	0.88
25	11	55	15	249	305	1	305	383	283	5655	0.93	0.93
										Average	0.81	0.89

Table II. Results for the Benavent et al. instances

Istance	$\|V\|$	$\|R\|$	L	UB_1	SEC_1	UB_2	SEC_2	UB_3	SEC_3	UB_3/UB_1	UB_3/UB_2
val1a	24	39	42	339	1	313	83	275	199	0.81	0.88
val1b	24	39	42	339	1	313	82	275	295	0.81	0.88
val1c	24	39	42	359	1	330	96	285	397	0.79	0.86
val2a	24	34	74	645	1	546	44	504	253	0.78	0.92
val2b	24	34	74	645	1	546	44	504	231	0.78	0.92
val2c	24	34	74	754	1	625	54	528	303	0.70	0.84
val3a	24	35	27	162	1	174	55	126	136	0.78	0.72
val3b	24	35	27	162	1	174	56	126	118	0.78	0.72
val3c	24	35	27	194	1	172	58	127	219	0.65	0.74
val4a	40	69	80	845	7	804	602	627	1534	0.74	0.78
val4b	40	69	80	845	5	804	594	627	1162	0.74	0.78
val4c	40	69	80	845	4	804	615	627	929	0.74	0.78
val4d	40	69	80	879	1	836	619	701	874	0.80	0.84
val5a	34	65	75	1711	6	1304	603	1039	4797	0.61	0.80
val5b	34	65	75	1711	4	1304	690	1039	3976	0.61	0.80
val5c	34	65	75	1711	3	1304	567	1039	3501	0.61	0.80
val5d	34	65	75	1711	2	1304	604	1153	1629	0.67	0.88
val6a	30	50	53	436	2	446	137	388	816	0.89	0.87
val6b	30	50	53	436	1	446	137	363	781	0.83	0.81
val6c	30	50	53	470	1	446	153	438	239	0.93	0.98
val7a	40	66	43	519	6	491	149	461	1080	0.89	0.94
val7b	40	66	43	519	5	487	697	461	850	0.89	0.95
val7c	40	66	43	539	2	535	698	461	856	0.86	0.86
val8a	29	63	71	854	5	797	669	664	1470	0.78	0.83
val8b	29	63	71	854	3	799	713	664	1204	0.78	0.83
val8c	29	63	71	884	1	843	742	716	574	0.81	0.85
val9b	50	92	46	837	19	779	3174	774	4303	0.92	0.99
val9d	50	92	46	837	5	895	3364	774	2026	0.92	0.86
									Average	0.78	0.85

6. Conclusion

In this paper we have presented a constructive heuristic and two tabu search procedures for the CLARPIF, a complex Arc Routing Problem in which vehicles may unload or replenish at intermediate facilities and the length of any route may not exceed a given upper bound. Computational results show that tabu search is able to provide substantial improvements over constructive heuristic solutions. Future research effort should concentrate on developing tight lower bounds.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Council under grant OPG0039682, by Ministero dell'Università e della Ricerca Scientifica (MURST), and by Consiglio Nazionale delle Ricerche (CNR). This support is gratefully acknowledged.

This research was partially supported by the Italian National Research Council, by the Center of Excellence on High-Performance Computing, University of Calabria, Italy, and by the Canadian Natural Sciences and Engineering Research Council under grants 227837-00 and OGP0039682. This support is gratefully acknowledged.

References

1. Assad, A. A. and Golden, B. L.: Arc routing methods and applications, in M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (eds), *Handbook of Operations Research and Management Science: Networks*, North-Holland, Amsterdam, 1995, pp. 375–483.
2. Beasley, J. E.: Route first–Cluster second methods for vehicle routing, *Omega* **11** (1983), 403–408.
3. Benavent, E., Campos, V., Corberán, A. and Mota, E.: The capacitated arc routing problem: Lower bounds, *Networks* **22** (1992), 669–690.
4. DeArmon, J. A.: A comparison of heuristics for the capacitated Chinese postman problem, Master Thesis, University of Maryland at College Park, 1981.
5. De Rosa, B., Ghiani, G., Improta, G. and Musmanno, R.: The arc routing and scheduling problem with transshipment, *Transport. Sci.* **36** (2001), 301–313.
6. Edmonds, J. and Johnson, E. L.: Matching, Euler tours and the Chinese postman problem, *Math. Programming* **5** (1973), 88–124.
7. Eglese, R. W. and Li, L. Y. O.: Efficient routing for winter gritting, *J. Oper. Res. Soc.* **43** (1992), 1031–1034.
8. Eiselt, H. A., Gendreau, M. and Laporte, G.: Arc routing problems, Part 1: The Chinese postman problem, *Oper. Res.* **43** (1995), 231–242.
9. Eiselt, H. A., Gendreau, M. and Laporte, G.: Arc routing problems, Part 2: The rural postman problem, *Oper. Res.* **43** (1995), 399–414.
10. Frederickson, G. N.: Approximation algorithms for some postman problems, *SIAM J. Comput.* **7** (1979), 178–193.
11. Ghiani, G., Hertz, A. and Laporte, G.: Recent algorithmic results for arc routing problems, E. Kozan and A. Ohuchi (eds), *Putting Theory into Practice*, Kluwer Acad. Publ., New York, 2001, pp. 1–20.

12. Ghiani, G., Improta, G. and Laporte, G.: The capacitated arc routing problem with intermediate facilities, *Networks* **37** (2001), 134–143.
13. Ghiani, G., Guerriero, F., Improta, G. and Musmanno, R.: Solving a complex public waste collection problem in Southern Italy, *Internat. Trans. Oper. Res.* (2004), forthcoming.
14. Glover, F. and Laguna, M.: *Tabu Search*, Kluwer Acad. Publ., New York, 1997.
15. Golden, B. L. and Wong, R. T.: Capacitated arc routing problems, *Networks* **11** (1981), 305–315.
16. Gendreau, M., Hertz, A. and Laporte, G.: A tabu search heuristic for the vehicle routing problem, *Management Sci.* **40** (1994), 1276–1290.
17. Hertz, A., Laporte, G. and Mittaz, M.: A tabu search heuristic for the capacitated arc routing problem, *Oper. Res.* **48** (2000), 129–135.
18. Jansen, K.: Bounds for the general capacitated routing problem, *Networks* **23** (1993), 165–173.
19. Lenstra, J. K. and Rinnooy Kan, A. H. G.: On general routing problems, *Networks* **6** (1976), 273–280.
20. Lemieux, P. F. and Campagna, L.: The snow plowing problem solved by a graph theory algorithm, *Civil Eng. Systems* **1** (1984), 337–341.
21. Taillard, É. D.: Robust taboo search for the quadratic assignment problem, *Parallel Comput.* **17** (1991), 433–445.
22. Wirasinghe, S. C. and Waters, N. M.: An approximate procedure for determining the number, capacities and locations of solid waste transfer-stations in an urban region, *European J. Oper. Res.* **12** (1983), 105–111.