15th Triennial World Congress, Barcelona, Spain

GENERAL ARC ROUTING PROBLEMS SOLVED BY A CUTTING PLANE ALGORITHM AND A GENETIC ALGORITHM

P. Lacomme[(1)], C. Prins[(2)], W. Ramdane-Chérif[(2)]

[(1)]*University Blaise Pascal, Laboratory for Computer Science (LIMOS)*
Campus Universitaire des Cézeaux, 63177 Aubière Cedex, France
Mail : lacomme@sp.isima.fr

[(2)]*University of Technology of Troyes, Laboratory for Optimization of Industrial Systems*
12 Rue Marie Curie, BP 2060, 10010 Troyes Cedex, France
Mail : {prins, ramdane}@utt.fr

Abstract: This paper considers an extended version of the Capacitated Arc Routing Problem (E-CARP), obtained by adding realistic constraints like prohibited turns to the basic CARP. Two integer linear models are presented, a mono-objective version and a bi-objective one. A cutting plane method for the first version provides either an optimum or a lower bound that are used to benchmark a genetic algorithm on 11 CARP instances from the literature, enriched by additional constraints. The results prove the GA is nearly optimal for these networks with up to 16 nodes and 52 arcs. *Copyright © 2002 IFAC*

Keywords: transportation, linear programming, arc routing, genetic algorithm

1. INTRODUCTION

The basic *Capacitated Arc Routing Problem* (*CARP*) is defined in the literature on an undirected network $G = (V, E)$ with a set V of n nodes and a set E of m edges. A fleet of identical vehicles of capacity Q is based at a depot node s. A subset R of *required edges* must be serviced by a vehicle. All edges can be traversed any number of times. Each edge (i, j) has a traversal cost $c_{ij} \geq 0$ and a demand $r_{ij} \geq 0$.

The CARP consists of determining a set of vehicle trips of minimum total cost, such that each trip starts and ends at the depot, each required edge is serviced by one single trip, and the total demand handled by any vehicle does not exceed Q. The cost of a trip comprises the costs of its serviced edges and of its intermediate connecting paths. Many applications occur in road networks: urban waste collection, snow removal, sweeping, gritting, etc. Demands are usually amounts to be collected along the streets (urban waste) or delivered (salt in winter). Costs are often distances or travel times.

The undirected version concern roads whose both sides can be serviced during one traversal and in any direction, a common situation in quiet suburban areas. A directed version is sometimes studied: each arc is one street (or one side of street) with an imposed direction for service. Note the difference between an edge (i, j) and two opposite arcs (i, j) and (j, i): both represent a 2-way street but each is serviced separately for the pair of arcs. The undirected and directed versions are NP-hard, even in the single-vehicle case called *Rural Postman Problem*.

In applications like urban waste collection, the street network is obviously *mixed* (with edges *and* arcs) and some turns are not allowed. Moreover, the traversal of a street for service takes longer than a deadheading traversal, and speed limits can give different times for the two directions of an edge. The E-CARP denotes a CARP with the following complications:

- mixed network
- each arc or edge (i, j) has a deadheading cost c_{ij} distinct from its servicing cost w_{ij}
- prohibited turns.

2. INTEGER LINEAR FORMULATION

Our formulation generalises a model proposed by Golden and Wong in 1981 for the basic CARP. Even this early model has never been evaluated, due to the lack of solvers powerful enough at that time.

2.1 Input data

The mixed network is coded as a directed multigraph $G = (V, A)$. V is a set of n nodes with a depot at node 1. A is a set of m arcs with each edge coded as two opposite arcs. Parallel arcs may exist from a node i to a node j, so arc indexes from 1 to m are used instead of the ambiguous notation (i, j). Each arc u has a traversal cost c_u and a demand q_u. We assume the required arcs are the ones with non-zero demands. Such arcs have also a servicing cost w_u. A fleet of K identical vehicles of capacity Q is based at the depot. All costs, demands and capacities are non-negative real numbers. No demand exceeds Q, *i.e.* $Q \geq \max \{q_u \mid u \in A\}$. For tackling the mixed graph and prohibited turns, we need also for each arc $u = (i,j)$:

- $S(u)$ the set of adjacent arcs $v = (j,k)$ that a vehicle may traverse immediately after u,
- $P(u)$ the set of adjacent arcs $v = (k,i)$ that a vehicle may traverse immediately before u,

- $Inv(u)$ the index of the opposite arc.

$S(u)$ and $P(u)$ are useful to describe many types of constrained turns that can correspond to a road sign, an excessive turning circle, a too narrow street: if for any reason it is forbidden to turn from arc u to arc v, v is removed from $S(u)$ and u from $P(v)$. $Inv(u)$ is required to distinguish between a genuine arc and a pair of arcs coding an edge: if two arcs u and v represent the same edge, then $Inv(u) = v$ and $Inv(v) = u$. If u and v are two opposite arcs requiring service separately, then $Inv(u) = Inv(v) = 0$.

2.2 Decisional variables

- $x_{u,v}^p$ number of times vehicle p traverses arc v immediately after arc u
- l_u^p binary variable equal to 1 iff arc u is serviced by vehicle p.

The initial model proposed by Golden and Wong uses binary variables x_{up} equal to 1 if and only if vehicle p traverses arc u. These variables are binary thanks to a theorem for the undirected CARP, stating that *there exists an optimal solution in which no trip traverses the same edge more than once, in a given direction*. Unfortunately, this property does not hold for directed or mixed graphs and the variables become integral.

Moreover, the x variables now require two subscripts to handle prohibited turns correctly. They need to be defined only for permitted pairs of consecutive arcs (u,v). In real road networks, each arc (street) is followed by 4 arcs on average, including a possible U-turn. There are then $4m$ permitted pairs on average, even less in case of prohibited turns. The variables l can be defined only for the r required arcs. In practice, the total number of variables is $K(4m+r)$.

2.3 The E-CARP model

$$(1)\quad \text{Min} \sum_{p=1}^{K} \sum_{v \in R} w_v \times l_v^p + \sum_{p=1}^{K} \sum_{v \in R} \sum_{u \in P(v)} c_v \times \left(x_{u,v}^p - l_v^p \right) + \sum_{p=1}^{K} \sum_{v \in A-R} \sum_{u \in P(v)} c_v \times x_{u,v}^p$$

subject to:

$$(2)\quad \forall p = 1 \ldots K, \forall u \in A: \sum_{v \in P(u)} x_{v,u}^p = \sum_{v \in S(u)} x_{u,v}^p$$

$$(3)\quad \forall u \in R, Inv(u) = 0: \sum_{p=1}^{K} l_u^p = 1$$

$$(4)\quad \forall u \in R, Inv(u) > 0: \sum_{p=1}^{K} \left(l_u^p + l_{Inv(u)}^p \right) = 1$$

$$(5)\quad \forall p = 1 \ldots K, \forall u \in R: \sum_{v \in S(u)} x_{u,v}^p \geq l_u^p$$

$$(6)\quad \forall p = 1 \ldots K: \sum_{u \in R} l_u^p \times q_u \leq Q$$

$$(7)\quad \forall p = 1 \ldots K, \forall i = 2 \ldots n: \sum_{\substack{u \in A \\ b(u)=i}} f_u^p - \sum_{\substack{u \in A \\ e(u)=i}} f_u^p = \sum_{\substack{u \in R \\ b(u)=i}} l_u^p$$

$$(8)\quad \forall p = 1 \ldots K, \forall u \in A: f_u^p \leq n^2 \times \sum_{v \in S(u)} x_{u,v}^p$$

$$(9)\quad \forall p = 1 \ldots K, \forall u \in A, \forall v \in S(u): x_{u,v}^p \in \mathbb{N}$$

$$(10)\quad \forall p = 1 \ldots K, \forall u \in R: l_u^p \in \{0,1\}$$

$$(11)\quad \forall p = 1..K, \forall u \in A: f_u^p \geq 0$$

The three main terms in the objective function (1) correspond to the total servicing cost of the required arcs, the total traversal cost of the required arcs, and the total traversal cost of non-required arcs. In each flow constraint (2), the number of times vehicle k enters arc u must be equal to the number of times it leaves u. In the example of figure 1, the values on each arc are the number of traversals by vehicle k. We see that arc u is entered and left four times.

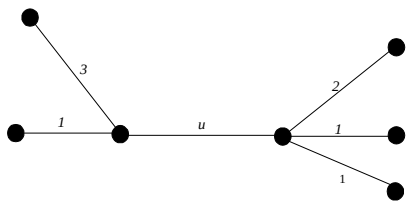

Fig. 1. Flow constraints on arc u.

Constraints (3) mean that each required true arc (*i.e.*, not an edge) of the original network must be serviced, and by one single vehicle. Constraints (4) concern required edges. When two arcs u and $Inv(u)$ code the same edge, only one of them must be collected, and by one single vehicle. The constraints (5) prevent the required arcs from being serviced by a vehicle when the vehicle does not traverse them.

Constraints (6) limit the total amount of demand serviced by one vehicle to its capacity. Constraints (7)-(8) prevent *subtours*, *i.e.* invalid cycles not connected to the depot. Contrary to the Travelling Salesman Problem, subtours may cross a node more than once and require more complex elimination constraints. In fact, we have extended the constraints proposed by Golden and Wong to tackle the mixed graph and prohibited turns. The total number of constraints (without positivity and bound constraints) is $K(2m+n+1)+r(K+1)$. For street networks, we have $m \approx 4n$ and $K(9n+1)+r(K+1)$ constraints.

2.4 Alternative subtour elimination constraints

Subtours can also be eliminated with constraints (7'). They prevent the formation of subtours for any non-empty subset of nodes E that does not contain the depot. Due to the huge number of subsets ($2^{n-1}-1$),

such a formulation cannot be handled directly by a linear programming software.

$$\forall p = 1 \ldots K, \forall E \subseteq \{2,3,\ldots,n\}, E \neq \varnothing :$$

$$\sum_{i \in E} \sum_{j \in E} x_{ij}^p - n^2 . y_{1e}^p \leq |E| - 1$$

$$(7') \qquad \sum_{i \in E} \sum_{j \notin E} x_{ij}^p + y_{2e}^p \geq 1$$

$$y_{1e}^p + y_{2e}^p \leq 1$$

$$y_{1e}^p, y_{2e}^p \in \{0,1\}$$

3. A BI-OBJECTIVE EXTENSION

In municipal waste collection, the nominal capacity specified by the manufacturer can be considered as a soft constraint for vehicles with compactors. Depending on the kind of waste, overloads reaching 10% are possible. Therefore, solutions with overloaded trips must not be excluded since they can be acceptable in practice. This section shows how to adapt the previous model for tackling two objectives: the overloads and the total cost of the trips.

3.1 Bi-objective optimization

The objective function to be minimized is now $H = \alpha_1 . H_1 + \alpha_2 . H_2$ where:

- H_1 is the total or the maximal overload
- H_2 is the total cost of the trips
- α_2 and α_1 are given weights representing the relative importance of H_1 and H_2.

Obviously, the main objective in practice is to find a capacity-feasible solution or, if not possible, one with minimal overloads. In the set of capacity-feasible solutions, the objective is to achieve the minimum total cost. If capacity-feasible solutions do not exist or cannot be computed in an acceptable computational time, priority must be given to minimizing overload.

This hierarchical bi-objective optimization problem can be solved by setting the values of α_1 and α_2 such as $\min(\alpha_1 . H_1) > \max(\alpha_2 . H_2)$. For instance, if the total cost is always smaller than 1000 and if overloads are integers, well scaled values of *H* are obtained with $\alpha_1 = 1000$ and $\alpha_2 = 1$. An objective function value of 11071 means a total overload equal to 11 and a solution cost equal to 71.

3.2 Minimizing the total overload

The model of section 2 is in fact the version reduced to objective H_2. We just present the modifications required in the objective function and in the constraints for tackling H_1. The overload *Extra*(*p*) of vehicle *p* and the total overload *T* can be defined as:

$$Extra(p) = \sum_{u \in R} l_u^p \times q_u - Q$$

$$T = \sum_{p=1}^{K} \max(0, Extra(p))$$

A linear formulation is possible by defining *Extra*(*p*) as a new positive variable for each vehicle *p* and by rewriting constraints (6) as follows:

$$(6') \quad \forall p = 1 \ldots K : \sum_{u \in R} l_u^p \times q_u \leq Q + Extra(p)$$

The new bi-objective function is obtained by multiplying (1) by α_2 and by adding the weighted sum of overloads:

$$\alpha_1 \times \sum_{p=1}^{K} Extra(p)$$

3.3 Minimizing the maximum overload

The maximal vehicle overload is: $\underset{p}{Max}(Extra(p))$.

It can be minimized by replacing constraints (6) by (6') like in 3.2 and by defining a positive variable *Z* playing the role of an upper bound on overloads. The overloads are actually bounded using constraints (6''):

$$(6'') \quad \forall p = 1 \ldots K : Extra(p) \leq Z$$

The bi-objective function is obtained by multiplying (1) by α_2 and by adding $\alpha_1 \times Z$: at the optimum, *Z* will be equal to the maximum overload.

4. CUTTING PLANE ALGORITHM

4.1 Introduction

Commercial solvers can solve only small instances of the integer linear models proposed in section 2, especially the version (denoted *P* in the following) with the exponential number of subtour elimination constraints. However, a huge number of constraints does not mean they are all active in an optimal solution. This allows an iterative approach whose the principle was first introduced by Dantzig (Dantzig *et al.*, 1954) for the travelling salesman problem.

The idea is to try to solve a relaxation *P*' of *P* without subtour elimination constraints. The constraints of *P* violated in the optimal solution of *P'* are added to *P'*. Such linear constraints added to the program at each iteration are called *cutting planes*. The process is iterated until the optimum found for *P'* is feasible for *P*. Such an optimum is also optimal for *P*. A cutting plane algorithm is expected to reach the optimal solution using only a partial description of *P*. However, in the worst case, it can perform an exponential number of iterations if all constraints initially discarded need to be reintroduced.

4.2 Cutting plane algorithm (CPA) for the E-CARP

We need to define for the cutting-plane algorithm:

- the constraints defining P' (relaxation of P)
- a procedure identifying the constraints of P violated in an optimal solution S for P'.

The relaxation of P. Initially, P' is the integer linear program obtained by removing from P all subtour elimination constraints (see section 2).

Identifying subtours in S. A valid solution S to P has up to K cycles, all connected to the depot. Invalid solutions contain subtours. The set X of subtours can be built with a graph search algorithm (see for instance Cormen *et al.*, 1990, for an implementation).

Breaking subtours. Any subtour in X is performed by a vehicle k on a subset E of nodes. It will not reappear in the subsequent resolutions of P' if we add to P' the constraints (7') for subset E and vehicle k.

4.3 Outline of CPA

Figure 2 gives an outline working directly on P (no separate variable for P') and based on three functions:

- *Solve*(P) solves an integer linear program P to optimality. Any commercial or freeware linear programming package can be used.
- *Subtours*(S) runs a graph search algorithm on the solution S of P to return its set X of subtours.
- *Alter*(P,X) adds to P the constraints to forbid all subtours of X in subsequent iterations.

The CPA performs its iterations until:

- S has no subtour (S is then optimal for P).
- a stopping criterion holds (*e.g.* maximal number of iterations or computational time).

This algorithm can solve small instances optimally. It can be time-consuming for larger instances and only lower bounds can be obtained. Even in that case, the CPA is useful to evaluate heuristics.

Cutting plane algorithm (CPA)

Input:
1. P the E-CARP problem

Output:
1. S the E-CARP solution

External functions:
1. *Solve*(P) returns the integer solution of P
2. *SubTours*(S) returns the set of subtours of S that do not contain the depot
3. *Alter*(P,X) adds to P the required constraints to forbid subtours in X

Local variables:
1. X the set of subtours not connected to the depot

Algorithm:
1. Remove subtour elimination constraints from P
2. **While** not stop **Do**
   2.1. $S \leftarrow$ *Solve*(P)
   2.2. $X \leftarrow$ *SubTours*(S)
   2.3. $P \leftarrow$ *Alter*(P,X)
3. **End Do**
4. **Return** S

Fig. 2. Cutting plane algorithm for the E-CARP

5. A LIBRARY FOR THE E-CARP

Two sets of instances were proposed for the basic CARP by DeArmon (1981) and by Belenguer and Benavent (1997). All edges in these instances require service. The files can be downloaded via Internet at address: *ftp://matheron.estadi.uv.es/pub/CARP*. Since no such benchmarks are available for our E-CARP, we use some DeArmon's instances and convert them into E-CARP instances by adding new constraints. A library composed of 11 instances based on DeArmon's files gdb1 to gdb5 and gdb14 to gdb19 is proposed in (Lacomme *et al.*, 2001b). The modified files have a name ending with "e" *e.g.*, gdb1 gives the E-CARP instance gdb1e. They can be obtained by sending an e-mail to the authors.

6. COMPUTATIONAL EVALUATION OF CPA

6.1 Results

The testing is done on a 600 MHz PC (Windows 95), using the Xpress-MP linear programming software (*http://www.dashoptimization.com*). Our program is written in Delphi 5 and calls Xpress for adding constraints. Table 1 shows the results. The variables and constraints concern the first relaxed program P'_1. Six problems are solved to optimality (asterisks). In the other cases (in brackets), no integer solution is obtained and the costs in brackets correspond to the continuous relaxation of P'_1. Such values are still useful as lower bounds for heuristic algorithms.

Table 1. Solutions obtained with the cutting plane algorithm CPA

Name	Nodes x arcs	Variables	Constraints	Solution cost
Gdb1e	12x44	1441	998	311*
Gdb2e	12x52	2473	1766	(379)
Gdb3e	12x44	1486	1123	293*
Gdb4e	11x38	937	742	297*
Gdb5e	16x52	2257	1664	(422.5)
Gdb14e	7x42	2311	1001	135*
Gdb15e	7x42	2153	1090	44*
Gdb16e	8x56	3561	2270	(119)
Gdb17e	8x56	3561	2265	(90)
Gdb18e	9x72	5901	2421	(87)
Gdb19e	8x22	481	333	69*

6.2 Example of solution details for problem gdb19e

Figure 3 shows the network in which each arc has an index and (in brackets) a deadheading cost, a service cost and a demand. Vehicle capacity is 27. In figures 4 to 6, the arcs labelled v_k are traversed by vehicle k and dashed lines are deadheading arc traversals. In the sequel P'_i denotes the relaxation of P at iteration i.

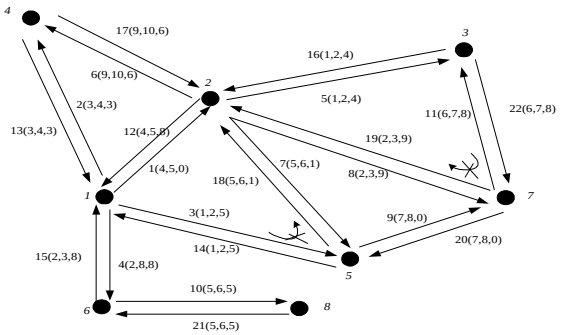

Fig. 3. Network of problem Gdb19e

Iteration 1. The solution (figure 4) costs 69 and contains two subtours with arcs (7,3), (3,7) and (6,8), (8,6). Constraints for breaking these subtours are added to P'_1 giving P'_2.

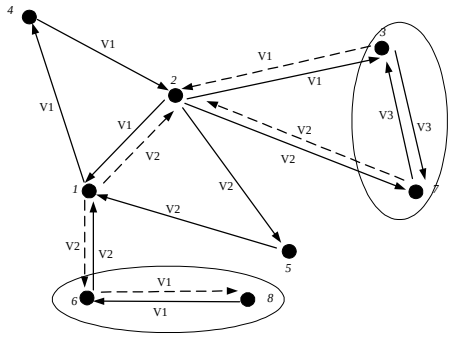

Fig. 4. First solution with two subtours

Iteration 2. The solution of figure 5 costs 69 and contains one subtour (2,3), (3,7), (7,3), (3,2). P'_2 is modified to definitely prevent this subtour.

Iteration 3. The resolution gives a solution of cost 69 without subtours (figure 6). This solution is also optimal for P and the algorithm may be stopped.

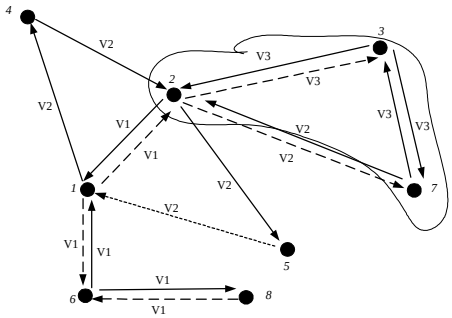

Fig. 5. Solution of P'_2

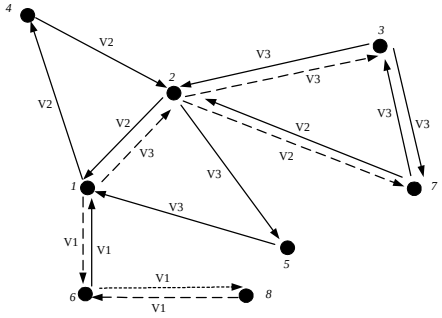

Fig. 6. Optimal solution for gdb19e

This solution has 3 trips with the following node lists:

- vehicle 1: 1,6,8,6,1
- vehicle 2: 1,4,2,7,2,1
- vehicle 3: 1,2,3,7,2,5,1

The evolution of the objective function during the resolution process is indicated below.

- Resolution of P'_1: the branch-and-bound algorithm of Xpress finds two integer solutions costing 71 and 69, the last one with two subtours.
- Resolution of P'_2: Xpress finds again two integer solutions with costs 71 and 69, but this time there is only one subtour in the optimal solution.
- Resolution of P'_3: Xpress finds four successive integer solutions with costs 77, 75, 71 and 69. The algorithm stops because the last one has no subtour: it is also an optimal solution to P.

7. THE HYBRID GENETIC ALGORITHM

Lacomme, Prins and Ramdane-Chérif have designed a very efficient hybrid genetic algorithm (HGA) for the basic undirected CARP (Lacomme *et al.*, 2001a). This HGA outperforms the best metaheuristics previously published, including a sophisticated tabu search (Hertz *et al.*, 2000), and breaks several open instances of the literature. The optimality can be proven on many instances, thanks to a tight lower bound designed by Belenguer and Benavent (1997).

Roughly speaking, each chromosome used by the HGA is a solution coded as a sequence of arc indexes, without trip delimiters. Such a sequence is evaluated by a tour splitting algorithm that cuts it optimally into trips. The HGA searches the sequence space to find the optimal sequence, *i.e.* the one giving an optimal CARP solution when evaluated.

The HGA works with a small population of distinct solutions. The solutions obtained by three good heuristics are included: Path-Scanning, Augment-Merge (both proposed by Golden and Wong, 1981) and Ulusoy's heuristic (Ulusoy, 1985). Children are generated using the classical OX crossover. The GA is hybrid because a local search is used as a mutation operator. The efficiency comes from the high mutation rate (10 to 20%). Such rates are possible without a premature convergence of the GA, thanks to the distinct solutions.

We have extended the HGA for tackling the E-CARP. For instance, forbidden turns are tackled by precomputing an arc-to-arc distance matrix D. $D(u,v)$ is the minimal cost of the paths from arc u (excluded) to arc v (excluded), taking into account forbidden turns. This matrix can be computed by adapting Dijkstra's algorithm (Cormen, 1990).

Table 2. Comparison between CPA and HGA costs

Name	CPA cost	HGA cost	Deviation %
Gdb1e	311*	335	7.71
Gdb2e	(397)	400	0.76
Gdb3e	293*	293*	0
Gdb4e	297*	297*	0
Gdb5e	(422.5)	451	6.74
Gdb14e	135*	135*	0
Gdb15e	44*	44*	0
Gdb16e	(119)	129	8.40
Gdb17e	(90)	91	1.11
Gdb18e	(87)	111	27.59
Gdb19e	69*	69*	0

Table 2 compares the cutting plane algorithm CPA with the HGA on the 11 E-CARP instances. The HGA uses a population of 30 solutions, a mutation rate of 10%, and stops after 40,000 crossovers. It retrieves 5 optimal solutions. Its average deviation from CPA costs is 4.75%. Note that HGA is perhaps also optimal when CPA provides just a lower bound (*e.g.* Gdb2e). The CPU time per instance is less than 2 minutes on a 600 MHz PC with Windows 95.

8. FINAL COMMENTS

This paper presents an extended version of the CARP and two integer linear programs, one for the single objective case, the other one for a bi-objective version. The first model is exploited by a cutting plane algorithm tested on a set of 11 instances. The cutting plane is able to solve instances up to 44 arcs to optimality. The hybrid GA proposed by (Lacomme *et al.*, 2001a) is enhanced for solving the E-CARP. The HGA is very efficient since it retrieves all the optimal solutions found by the CPA and displays on larger instances small deviations from the lower bound provided by the CPA. Several extensions to this work are already envisaged, for instance:

- enhancing the CPA for solving larger instances
- calling the CPA in the HGA to solve sub-problems optimally.
- using a truncated HGA to provide tight upper bounds in a branch-and-cut algorithm.

REFERENCES

Belenguer J.M. and E. Benavent (1997). A Cutting plane algorithm for the capacitated arc routing problem. *Research Report*, Dept. of Statistics and OR, Univ. of Valencia (Spain). To appear in *Computers and Operations Research*.

Cormen T.H., C.E. Leiserson and R.L. Rivest (1990). *Introduction to algorithms*. The MIT Press.

Dantzig G.B., D.R. Fulkerson and S.M. Johnson (1981). Solution of a large scale traveling salesman problem. *Operations Research*, **2**, 393-410.

DeArmon J.S. (1981). A comparison of heuristics for the capacitated chinese postman problem. *Master's Thesis*, The University of Maryland at College Park, MD, USA.

Golden B.L. and R.T.Wong (1981). Capacitated arc routing problems. *Networks*, **11**, 305-315.

Hertz A., G. Laporte and M. Mittaz (2000). A tabu search heuristic for the capacitated arc routing problem, *Operations Research*, **48**, 129-135.

Lacomme P., C. Prins and W. Ramdane-Chérif (2001a). A genetic algorithm for the Capacitated Arc Routing Problem and its extensions. In: *Applications of evolutionary computing* (E.J.W. Boers, Ed.), 473-483, *Lecture Notes in Computer Science* 2037, Springer.

Lacomme P., C. Prins and W. Ramdane-Chérif (2001b). An integer linear model for general arc routing problems, *European Simulation Symposium* (ESS'01), Marseille, France, 18-20 October.

G. Ulusoy (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, **22**, 329-337.