A Genetic Algorithm with Population Management (GA|PM) for the CARP

Christian Prins* Marc Sevaux† Kenneth Sörensen‡

*University of Technology of Troyes
LOSI, BP 2060, 12 rue Marie Curie
F-10010 Troyes, France
Christian.Prins@utt.fr

†University of Valenciennes
LAMIH-SP, Le Mont-Houy
F-59313 Valenciennes, France
Marc.Sevaux@univ-valenciennes.fr

‡University of Antwerp
TEW, Prinsstraat 13
B-2000 Antwerpen, Belgium
Kenneth.Sorensen@ua.ac.be

Abstract

In this paper, we apply the framework of genetic algorithms with population management (GA|PM) to the capacitated arc routing problem (CARP). GA|PM use distance measures to control the diversity of a small population of high-quality solutions. The algorithm is compared on the 23 GDB instances from Golden, DeArmon and Baker. The GA|PM is able to retrieve the same solutions much faster than an improved memetic algorithm, and can therefore be considered the best method to date.

1 The Capacitated Arc Routing Problem

The *Capacitated Arc Routing Problem* (CARP) is a vehicle routing problem raised by applications like urban refuse collection. It is defined 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 W is based at a depot node. Each edge e can be traversed any number of times, each time with a cost c_e, and has a non-negative demand q_e. All costs and demands are integers. The τ edges with non-zero demands, called *tasks*, require service by a vehicle. The goal is to determine 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 W.

The CARP is $\mathcal{NP}$-hard and exact methods are still limited to small instances with at most 25 edges. Hence, larger instances must be solved in practice with heuristics. Among fast constructive methods, one can cite for instance Path-Scanning or PS [5], Augment-Merge or AM [6] and Ulusoy's heuristic or UH [12]. Metaheuristics available are very recent and include for instance tabu search [7] and guided local search [2]. The most effective algorithm is [9] an improved version of a memetic algorithm (MA) presented by Lacomme, Prins and Ramdane-Chérif at EURO-GP 2001 [8]. All these heuristics can be evaluated thanks to very good lower bounds [1].

2 Principles of GA|PM

GA|PM (Genetic Algorithm with Population Management) is a new template of GA introduced by Sörensen [11]. It is structured much like a standard GA, but differs in the use of population management and local search. An outline is given in algorithm 1.

Algorithm 1 – GA|PM outline

```
1: initialise population P
2: set population diversity parameter δ
3: repeat
4:    select: p1 and p2 from P
5:    crossover: p1 ⊗ p2 → c1, c2
6:    local search: on c1 and c2
7:    for each child c do
8:       if d_P(c) ≥ δ then
9:          remove solution: P ← P\b
10:         add solution: P ← P ∪ c
11:      end if
12:   end for
13:   update diversity parameter δ
14: until stopping criterion satisfied
```

GA|PM controls the diversity of a small population P of high-quality solutions. It requires a distance measure d that determines for each pair of solutions their relative distance (or similarity) in the solution space. Ideally, d should be a metric. Using this distance measure between solutions, the distance of a child-solution c to the population can be defined as:

$$d_P(c) = \min_{s \in P} d(c, s) \tag{1}$$

A child-solution c may be added to P if its distance to the population is greater than or equal to the current value of the *diversity parameter* δ (Equation 2). Otherwise, solution c is discarded. Another possibility mentioned by Sörensen is to mutate c until it satisfies Equation 2. The diversity parameter is dynamically adjusted to intensify or diversify the search. Several distances and control policies for δ can be found in [11].

$$d_P(c) \geq \delta \tag{2}$$

Like GA|PM, *memetic algorithms* [10] include a local search procedure but they have no population management. Other approaches that use a small population of high quality solutions include *scatter search* and *path relinking* [4]. GA|PM offers the advantage of being closer to classical GA in terms of algorithm structure and therefore considerably easier to implement.

3 A GA|PM for the CARP

The GA|PM developed in this section uses several components of the effective memetic algorithm proposed by Lacomme *et al.* [9] for the CARP. The common parts are described in section 3.1. In section 3.2, we discuss the population management features of the GA|PM.

3.1 The existing memetic algorithm

Solution encoding The network is coded as a symmetric digraph, in which each edge is replaced by two opposite arcs. A chromosome is an ordered list of the τ tasks, in which each task may appear as one of its two directions. Implicit shortest paths are assumed between successive tasks. The chromosome does not include trip delimiters and can be viewed as a giant tour for an uncapacitated vehicle. A procedure *Split* optimally partitions (subject to the sequence) the giant tour into feasible trips. The guiding function of the MA is the total cost of the resulting CARP solution.

Initialization The small population P of (typically) $nc = 30$ chromosomes is initialized with the solutions of the three CARP heuristics cited in introduction (PS, AM and UH), completed by random permutations. The MA forbids clones (identical chromosomes) by using a simple policy: at any time, P must contain solutions with distinct costs.

Selection and crossover At each iteration, two parents are selected by binary tournament and reproduce according to a slightly modified version of the classical order crossover (OX). One child is randomly selected, the other is discarded.

Local search The mutation is replaced by a local search procedure *LS*, called with a fixed probability p_{ls}, which works on the individual routes computed by *Split* instead of the giant tour. The moves include the removal of one or two consecutive tasks from a route, with reinsertion at another position, the exchange of two tasks, and 2-opt moves. All moves may involve one or two routes and the two traversal directions of an edge are tested in the reinsertions. Each iteration scans all these moves to perform the first improving move. *LS* stops when no more improvement can be found. The trips are then concatenated into a chromosome, which is re-evaluated by *Split* because this brings sometimes a further improvement.

Population update The resulting child replaces one of the $nc/2$ worst chromosomes, randomly chosen in the population and such that no clone is created. The MA stops after a given number of iterations ni, after a stabilization period of ns iterations without improving the best solution or when reaching a lower bound LB.

Restarts When the lower bound is not reached, nr short restarts are performed with an intensified local search (rate $p_{lsr} > p_{ls}$). Each restart begins with a special procedure which replaces ncr chromosomes by new ones ($ncr < nc$), while preserving the best solution. It stops when the lower bound is achieved or after a given number of iterations nir ($nir < ni$).

3.2 Population management added

Distance measure Distances taking into account the partition of tasks into trips, like the one proposed by Sörensen [11] for the VRP, give good results but are time-consuming. The best results were obtained with a distance without trip delimiters that generalizes the distance for R-permutations proposed by Campos et al. [3]. Consider two chromosomes with τ tasks for the CARP, S and T. Let S_i denote the i-th task of S and $inv(S_i)$ the other possible traversal direction of S_i. The classical distance between R-permutations counts the number of pairs of consecutive tasks $(S_i, S_{i+1}), i = 1, 2, \ldots \tau - 1$, which are preserved (not broken) in T.

CARP solutions exhibit a property called *reversal independence* by Sörensen: the reverse string represents the same solution. The GA|PM extends the distance for R-permutations to handle reversal independence: a pair (S_i, S_{i+1}) is counted as broken if neither (S_i, S_{i+1}) nor $(inv(S_{i+1}), inv(S_i))$ are found in T.

Control policy for the diversity factor Note that the distance takes its values in $[1, \tau-1]$. The diversity factor δ is initialized to 1. After π iterations without improving the best solution $P(1)$, it is multiplied by a growth factor γ. Each time the best solution is improved, δ is reset to 1. The growth of δ is stopped when it reaches a ceiling μ. In practice, μ is a fraction ϕ of the maximum distance value.

3.3 Resulting GA|PM algorithm

The overall structure of the GA|PM for the CARP is given by Algorithm 2. F denotes the objective function. The variable *stuck* counts the number of iterations with $\delta = \mu$. The algorithm stops when a given lower bound LB is reached, when the distance keeps its maximum value μ during σ iterations ($stuck = \sigma$) or when a maximum number of iterations ni is reached. The restarts have exactly the same structure.

4 Results

The GA|PM is compared to the best-known algorithm: the memetic algorithm of Lacomme et al. [9], described in 3.1 and called BMA in the sequel. The two algorithms are implemented in

Algorithm 2 – GA|PM algorithm for the CARP

1: $P\ leftarrow$ {the solutions of the CARP heuristics PS, AM and UH}
2: complete P using random chromosomes
3: sort P in increasing cost order
4: *cni*,*stuck* $leftarrow\ 0$
5: $\delta\ leftarrow\ 1$
6: $\mu\ leftarrow\ \phi \cdot (\tau - 1)$
7: **repeat**
8: $cni\ leftarrow\ cni + 1$
9: select two parents P_1, P_2 by binary tournament
10: apply crossover OX to P_1, P_2 and choose one child C at random
11: evaluate C with *Split*
12: **if** $random < p_{ls}$ **then**
13: improve C with the local search procedure LS
14: **end if**
15: draw k at random between $\lfloor nc/2 \rfloor$ and nc included
16: **if** $F(C) < F(P(1))$ or $d_{P \setminus P(k)}(C) \geq \delta$ **then**
17: $P(k)\ leftarrow\ C$
18: shift $P(k)$ to keep P sorted
19: **end if**
20: $\delta\ leftarrow\ \min(\mu, \delta \cdot \gamma)$
21: **if** $\delta = \mu$ **then**
22: $stuck\ leftarrow\ stuck + 1$
23: **else**
24: $stuck\ leftarrow\ 0$
25: **end if**
26: **until** $(F(P(1)) = LB)$ or $(stuck = \sigma)$ or $(cni = ni)$

Delphi and tested with and without restarts on the 23 GDB instances from Golden, DeArmon and Baker [5], for which 21 proven optima are known.

BMA is not a pure memetic algorithm: even if it does not use a genuine distance measure and a dynamic diversity control like GA|PM, it works on a small population and uses a simple mechanism (the distinct costs) to spread solutions. This why a standard memetic algorithm (SMA) is also included in the comparisons. SMA is derived from BMA by using a larger population, allowing clones and suppressing restarts.

The results are given in Table 1: a) the average deviation in % to the excellent lower bounds LB from Belenguer and Benavent [1], b) the worst deviation in %, c) the number of proven optima (LB reached), d) the average running time in seconds on a 1.8 GHz Pentium IV PC (Windows 2000), e) the number of instances requiring restarts (and, in brackets, the number of instances for which restarts lead to an optimum) and f) the average number of crossovers.

The BMA row corresponds to the MA as published in [9], with the following parameters: $nc = 30$, $p_{ls} = 0.1$, $ni = 20000$, $ns = 6000$, $nr = 20$, $ncr = 22$, $p_{lsr} = 0.2$ and $nir = 2000$. In other words, BMA performs a main phase of at most 20000 crossovers, with 10% of local search. If the given lower bound is not achieved, it executes up to 20 shorter restarts (max.

2000 crossovers), in which 22 solutions are replaced and the local search rate is doubled. The BMA-nr row concerns the same algorithm, but without restarts. The SMA row is obtained with populations of $nc = 60$ chromosomes.

The GAPM-nr row corresponds to GA|PM without restarts, with $p_{ls} = 0.5$, $\pi = 50$, $\gamma = 1.08$, $\phi = 0.5$ (so, $\mu = 0.5 \times (\tau - 1)$) and $\sigma = 1000$. If ϕ is increased, too many children are rejected and the algorithm spends too much time in unproductive iterations. The GAPM row provides the results when restarts are added, with $nr = 20$, $ncr = 28$ (the two best solutions are kept), $p_{lsr} = 0.5$ (local search rate is not modified) and $nir = 2000$. Various attempts with other parameters settings provide slightly degraded results.

The SMA is very fast but it provides the worst deviations and finds only 15 optima. Compared to the other GA without restarts (BMA-nr), the GA|PM decreases the average deviation to LB and finds two more optima. The average number of crossovers is divided by 3. The running time is only slightly improved, due to the overhead induced by distance computations and the increased local search rate.

The results for the GAs with restarts show that GA|PM can find the same results as BMA, but within one third of its running time. The average number of crossovers is practically divided by 5. GA|PM finds an optimum solution without restarts for 20 instances out of 23, versus 18 for BMA.

Table 1: Summary of results

Algorithm	Dev. LB	Worst dev.	LB hits	Av. time	Restarts	Av. Xovers
SMA	0.65	4.07	15	0.37	0	2750.5
BMA-nr	0.33	2.23	18	0.95	0	3013.1
GAPM-nr	0.24	2.23	20	0.90	0	880.9
BMA	0.17	2.23	21	4.79	5(3)	9960.2
GA|PM	0.17	2.23	21	1.59	3(1)	1968.9

5 Conclusions and future work

This case study on the CARP shows that the GAs with population management are relatively simple to implement and can retrieve all the optima found by the best solution method (a memetic algorithm), but in a shorter running time. The number of crossovers is much smaller, but the average time spent per crossover increases, due to the overhead induced by distance computation and greater local search rate.

In the future we plan on applying the GA|PM to two sets of larger CARP instances described by Belenguer and Benavent [1]. Application to other problems is another topic for future research.

References

[1] J.M. Belenguer and E. Benavent. A cutting plane algorithm for the Capacitated Arc Routing Problem. *Computers and Operations Research*, 30(5):705–728, 2003.

[2] P. Beullens, L. Muyldermans, D. Cattrysse, and D. Van Oudheusden. A guided local search heuristic for the Capacitated Arc Routing Problem. *European Journal of Operational Research*, 147(3):629–643, 2003.

[3] V. Campos, M. Laguna, and R. Martí. Context-independent scatter and tabu search for permutation problems. *To appear in INFORMS Journal on Computing*, 2003.

[4] F. Glover. *A Template for Scatter Search and Path Relinking*, volume 1363 of *Lecture Notes in Computer Science*, pages 1–53. Springer, Berlin, 1997.

[5] B.L. Golden, J.S. DeArmon, and E.K. Baker. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10(1):47–59, 1983.

[6] B.L. Golden and R.T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[7] A. Hertz, G. Laporte, and M. Mittaz. A tabu search heuristic for the Capacitated Arc Routing Problem. *Operations Research*, 48(1):129–135, 2000.

[8] P. Lacomme, C. Prins, and W. Ramdane-Chérif. A genetic algorithm for the Capacitated Arc Routing Problem and its extensions. In E.J.W. Boers et al., editor, *Applications of evolutionnary computing–EvoWorkshops 2001*, Lecture Notes in Computer Science 2037, pages 473–483. Springer, 2001.

[9] P. Lacomme, C. Prins, and W. Ramdane-Chérif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 2004. To appear.

[10] P. Moscato. Memetic algorithms: a short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New ideas in optimization*, pages 219–234. McGraw-Hill, 1999.

[11] K. Sörensen. *A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design*. PhD thesis, University of Antwerp, Belgium, 2003.

[12] G. Ulusoy. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22:329–337, 1985.