Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 35 (2008) 1725–1741
computers & operations research
www.elsevier.com/locate/cor

A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem

Abdellah El Fallahi, Christian Prins*, Roberto Wolfler Calvo

Institut Charles Delaunay, FRE CNRS 2848, Université de Technologie de Troyes, 12 Rue Marie Curie, BP 2060, 10010 Troyes, France

Available online 13 November 2006

Abstract

A generalization of the traditional vehicle routing problem (VRP) is studied in this paper. Each customer may order several products, the vehicles are identical and have several compartments, each compartment being dedicated to one product. The demand of each customer for a product must be entirely delivered by one single vehicle. However, the different products required by a customer may be brought by several vehicles. Two algorithms to solve this problem are proposed: a memetic algorithm with a post-optimization phase based on path relinking, and a tabu search method. These algorithms are evaluated by adding compartments to classical VRP instances.

Keywords: Vehicle routing problem; Memetic algorithm; Path relinking; Tabu search

1. Introduction

The vehicle routing problem (VRP) with several compartments (Multi-Compartment Vehicle Routing Problem or MC-VRP) is defined on an undirected network including one depot and a set of n customers. The depot stores m products which must be delivered to customers by a fleet of identical vehicles with m compartments of limited capacities. The compartment number p is reserved for a product p, $p = 1, 2, \ldots, m$. Each customer orders known amounts of one or more products. Each product required by a customer must be delivered by only one vehicle. However, the different products ordered can be brought by different vehicles. The MC-VRP consists of determining a set of routes satisfying all requests at minimal cost. In the proposed version, the number of vehicles (fleet size) is not imposed: it is a decision variable.

The MC-VRP is a generalization of the VRP, in which each vehicle has only one compartment and each customer is supplied by only one vehicle, see for instance Toth and Vigo [1] for a survey. Like its particular case the VRP, the MC-VRP is NP-hard. The possibility of multiple visits to the same customer characterizes also the VRP with split deliveries (split delivery VRP or SDVRP), introduced by Dror and Trudeau [2]. The SDVRP is a VRP in which customers can be supplied several times. The MC-VRP offers an intermediate level of splitting: the demand of a customer for one given product cannot be split, but his set of requested products can be delivered in several times.

Vehicle routing problems with compartments are widespread in logistics but seldom studied in research literature. The only published papers concern fuel distribution, e.g., Avella et al. [3], Brown and Graves [4], Brown et al. [5]

* Corresponding author.
E-mail address: christian.prins@utt.fr (C. Prins).

0305-0548/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2006.10.006

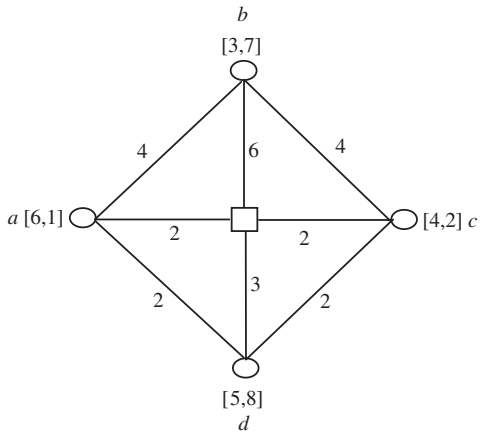

Fig. 1. Comparison between VRP, SDVRP and MC-VRP.

and Van der Bruggen et al. [6]. In this application, each compartment has its own capacity but can receive any fuel grade. Customers order large demands: most often, one compartment receives only one product for one customer. The main problem is to assign orders to tank trucks, to make the best use of fleet capacity. However, once this hard loading problem is solved, the routing part is rather easy: one TSP with very few stops must be solved separately for each truck.

The MC-VRP addressed in this paper is significantly different and, to the best of our knowledge, it is studied for the first time. The first assumption is that each compartment is dedicated to one product, in a broad sense. In fact, a "product" may represent different kinds of goods, but sharing common characteristics that require a specific compartment. Distribution to groceries is a good example, in which one cold compartment and one normal are used to deliver rather small orders. In that case, the two products are the kinds of food ordered, frozen or not. The second assumption is the possibility to bring the different products ordered by a customer using several vehicles. This option is frequent in inventory routing, when customers do not need to be present to receive their orders.

We found a genuine MC-VRP in the West of France, in the distribution of cattle food to farms. Even if compartments are almost identical in that case, sanitary rules recommend to always use the same for one species. For example, rabbits are very sensitive to viruses and it is too risky to transport granulates for this species in a compartment previously used for bovines. In a pickup context, the MC-VRP can be met in selective waste collection.

The differences between the VRP, the SDVRP and the MC-VRP are depicted in the small example of Fig. 1. There are four customers $\{a, b, c, d\}$, two products, one central depot and a fleet of three vehicles with capacity 18, divided into two compartments of size 9. The demands for each customer are indicated in square brackets and the traveling cost given on each edge. To obtain a VRP or SDVRP, it is assumed that the two products can be mixed and the two compartments merged into a single one: the demands for a, b, c, d, respectively, become 7, 10, 6 and 13.

The optimal VRP solution (cost 22) consists of three routes: (a, b), (c) and (d), with a total cost equal to 22. The optimal SDVRP solution has two routes (a, b, c) and (a, d) with total cost 19. Lastly, the MC-VRP requires two routes (a, b, c) and (a, d, c), with an intermediate total cost of 20 units. The first route delivers (6,0) to a, (3,7) to b and (0,2) to c. The second one brings (0,1) to a, (5,8) to d and (4,0) to c. Hence, a and c have a delivery for each product, while b and d receive their two products in one delivery.

Note that the problem still differs from the VRP if the two products must be delivered together, since to load the compartments a packing problem has to be solved. For example, consider two customers with the same request (2,1) and vehicles with two compartments of capacity 3: two routes are necessary for the MC-VRP whereas one is enough for the VRP.

The paper is organized as follows. Section 2 presents an integer linear programming model. Section 3 describes a memetic algorithm (MA), improved by a path relinking method in Section 4. Section 5 introduces a tabu search (TS) algorithm and its components. The instances used and the results obtained are discussed in Section 6.

2. Problem formulation

The MC-VRP is defined on a complete undirected network with a node set $N = \{0, 1, \ldots, n\}$ including one depot (node 0) and a set N' of n customers. Each edge (i, j) has a cost $c_{ij} = c_{ji}$ and the set of costs satisfies the triangle inequality. The depot stores a set $P = \{1, 2, \ldots, m\}$ of m products (in a broad sense), which must be delivered by a fleet $V = \{1, 2, \ldots, v\}$ of identical vehicles with m compartments. Compartment p of each vehicle is dedicated to product p and has a known capacity Q_p. Each customer i has a known request $q_{ip} \leqslant Q_p$ for each product p, possibly null for a product not ordered by the customer. In our version, there is also a maximum route length L. Then the MC-VRP can be formulated as follows:

$$\min \quad \sum_{i,j\in N} \sum_{k\in V} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in N} x_{ijk} \leqslant 1 \quad \forall j \in N', \forall k \in V, \tag{2}$$

$$\sum_{i\in N} x_{ijk} = \sum_{i\in N} x_{jik} \quad \forall j \in N', \forall k \in V, \tag{3}$$

$$\sum_{i,j\in S} x_{ijk} \leqslant |S| - 1 \quad \forall k \in V, \forall S \subseteq N', |S| \geqslant 2, \tag{4}$$

$$y_{jkp} \leqslant \sum_{i\in N} x_{ijk} \quad \forall j \in N', \forall k \in V, \forall p \in P, \tag{5}$$

$$\sum_{k\in V} y_{jkp} = 1 \quad \forall j \in N', \forall p \in P, \tag{6}$$

$$\sum_{j\in N'} y_{jkp} q_{jp} \leqslant Q_p \quad \forall k \in V, \forall p \in P, \tag{7}$$

$$\sum_{i,j\in N} c_{ij} x_{ijk} \leqslant L \quad \forall k \in V, \tag{8}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in N, \forall j \in N, i \neq j, \forall k \in V, \tag{9}$$

$$y_{jkp} \in \{0, 1\} \quad \forall j \in N', \forall k \in V, \forall p \in P, q_{jp} \neq 0. \tag{10}$$

The 0–1 variables x_{ijk} (9) are equal to 1 if and only if edge (i, j) is traversed by vehicle k. The 0–1 binary variables y_{jkp} (10) take value 1 if and only if customer j receives product p from vehicle k. The objective function (1) represents the total cost of the routes, to be minimized. In Eq. (2), each customer may be visited at most once by each route. Eq. (3) ensure the continuity of each route: if a vehicle enters node j, it must leave it. Eq. (4) are classical subtour elimination constraints. Constraints (5) set y_{jkp} to zero for each product p if customer j is not visited by vehicle k. Each product ordered by a customer is brought by one single vehicle, thanks to constraints (6). Constraints (7) and (8), respectively, prevent compartment capacity and route length violations.

The main interest of this model is to provide a compact problem specification: the problem is harder than the VRP and only very small instances can be solved by commercial MIP solvers. This is why metaheuristics are studied in the sequel to tackle the problem in practice.

3. Memetic algorithm

This section introduces the memetic algorithm proposed to solve the MC-VRP. A memetic algorithm (MA) is a genetic algorithm (GA) hybridized with a local search procedure used to intensify the search [7]. Here, the MA belongs to the *steady state* or *incremental* family: it works on one single population of constant size and each offspring obtained by crossover immediately replaces one existing solution. Such a GA structure is known for converging faster than *generational* versions, since a new best solution may reproduce immediately.

3.1. Algorithm overview

The proposed MA encodes solutions as chromosomes without trip delimiters, described in Section 3.2 and evaluated by a splitting procedure presented in 3.3. The MA starts from an initial population Π of σ well diversified solutions. The construction of Π, described in 3.4, is based on a heuristic separating the MC-VRP into several VRPs, one per product, and on a distance measure in solution space. The resulting population is sorted in ascending cost order.

During the MA iterations, the distance measure is replaced by the dispersal rule (11), in which $cost(X)$ is the cost of a solution X and Δ a given positive threshold. In other words, any two solutions S and T in Π may not have a cost difference smaller than Δ. It is true that two solutions with the same cost may have different structures, but the dispersal rule is a sufficient condition to avoid clones (identical solutions) and it is also simpler and faster than the distance: it is easily checked in $O(\log \sigma)$ by dichotomic search, if Π is kept sorted.

$$\forall S, T \in \Pi : |cost(S) - cost(T)| \geqslant \Delta. \tag{11}$$

Each MA iteration starts by randomly selecting two solutions P_1 and P_2, using the binary tournament method. One child-solution C is obtained with a modified OX crossover presented in Section 3.5. C undergoes with a given probability p_{LS} the local search described in 3.6. If C improves Π_σ the MA iteration ends by replacing one solution Π_k, k integer selected randomly in $[\sigma/2, \sigma]$, by child C, provided the cost spacing rule (12) holds. This rule is similar to (11), but Π_k, which is going to be replaced by C, is not concerned.

$$\forall S \in \Pi \backslash \{\Pi_k\} : |cost(S) - cost(C)| \geqslant \Delta. \tag{12}$$

However, to avoid missing a new best solution, the rule is overridden when the current best solution is improved, i.e., if $cost(C) < cost(\Pi_1)$. This exception is similar to the aspiration criterion in tabu search. Note that the *elitism* property of GAs holds: the best solution is either preserved or improved, but never degraded. If C violates rule (12) and is not a new best solution, it is simply discarded. The percentage of children discarded is tolerable (less than 10%), provided Δ is not too large. A rule of thumb is to set Δ to the average difference in cost between two solutions in a randomly generated population.

The MA stops after a maximum number of iterations (crossovers) α_{max} or after β_{max} iterations without improving the best solution. Its general structure is given by Algorithm 1. Note that no mutation operator is used. Indeed, provided the local search is not systematically applied to children, the cost spacing rule and the crossovers not followed by local search bring enough diversification.

Algorithm 1. General structure of the MA

```
split the MC-VRP into m VRPs and solve them using Clarke and Wright method
build an initial population Π of σ solutions using the VRP solutions (Section 3.4)
sort Π in ascending cost order
α, β = 0
repeat
   α = α + 1
   select two parents P1 and P2 in Π
   apply the crossover to P1 and P2 to get child C (Section 3.5)
   evaluate C with the splitting procedure (Section 3.3)
   apply local search to C with probability pLS (Section 3.6)
   if (C satisfies (12) and cost(C) < cost(Πσ)) or (cost(C) < cost(Π1)) then
      replace Πk by C, k selected randomly in [σ/2, σ]
      shift C to keep Π sorted
      if cost(C) < cost(Π1) then
         β = 0
      else
         β = β + 1
      end if
   end if
until (α = αmax) or (β = βmax)
```

```
Chromosome: 1  5  4  3  6  2
Customers : 1  3  2  2  3  1
Products  : 1  1  2  1  2  2
```

Fig. 2. Example of chromosome encoding.

3.2. Chromosome encoding

In [8], Prins describes an efficient MA for the VRP, based on chromosomes without route delimiters: each chromosome is defined as a node permutation S, which can be viewed as a giant tour performed by one vehicle of infinite capacity. The depot beginning and ending this tour and the shortest paths between successive nodes are implicit. This encoding is interesting because: (a) any chromosome S can be optimally partitioned into feasible routes using a splitting procedure, (b) any classical crossover for the Traveling Salesman Problem (TSP) can be reused and (c) the MA searches a smaller solution space. The giant tour approach was introduced by Beasley [9] to obtain a VRP solution from the solution of any TSP heuristic.

This encoding can be extended to the MC-VRP. Each customer i is represented by m genes (integers) ranging from $m(i-1)+1$, representing his demand for product 1, to mi, his request for product m. Hence, a gene S_i corresponds to customer $cust(S_i) = \lfloor (S_i + m - 1)/m \rfloor$ and to product $prod(S_i) = ((S_i - 1) \bmod m) + 1$. Let $\rho \leqslant nm$ be the number of demands $q_{ip} > 0$. To save space, only the genes corresponding to such demands are kept in chromosomes. Fig. 2 gives one example with $n = 3$, $m = 2$ and $\rho = nm$ non-null demands.

3.3. Chromosome evaluation

The splitting procedure used in [8] for the VRP partitions one chromosome S, given without trip delimiters, into feasible routes. It is based on an auxiliary graph $H = (X, A, Z)$. X contains $n+1$ nodes indexed from 0 to n, in which 0 is a fictitious node and any other node $i > 0$ represents customer S_i. The arc set A contains one arc (i, j) if the route visiting customers S_{i+1} to S_j (included) is feasible, i.e., if vehicle capacity and maximum route length are respected. The weight z_{ij} on the arc is the cost of the route. Thus, an optimal splitting of S corresponds to a minimum-cost path from node 0 to node n in H.

The procedure can be extended to the MC-VRP. In that case, X contains $\rho + 1$ nodes indexed from 0 onwards. Node 0 is still fictitious, while each other node i denotes the gene S_i that corresponds (see 3.2) to customer $cust(S_i)$ and to product $prod(S_i)$. The arc set A contains one arc (i, j), $0 \leqslant i \leqslant j \leqslant \rho$, if the trip serving the genes S_{i+1} to S_j is feasible, i.e., if the capacity of each compartment (13) and the maximum length (14) are respected. The weight z_{ij} of arc (i, j) is the cost of the associated trip. The optimal splitting of S corresponds to a minimum-cost path from node 0 to node ρ in H.

$$\forall p \in [1, 2, \ldots, m] : \sum_{k=i+1}^{j} q_{cust(S_k),prod(S_k)} \leqslant Q_p, \tag{13}$$

$$z_{ij} = c_{0,cust(S_{i+1})} + \sum_{k=i+1}^{j} c_{cust(S_k),cust(S_{k+1})} + c_{cust(S_j),0} \leqslant L. \tag{14}$$

Fig. 3 provides an example with $n = 4$ customers $\{a, b, c, d\}$, $m = 2$ products and two compartments with identical capacities $Q_1 = Q_2 = 9$. For instance, a_1 and a_2 denote the two genes for customer a, with demands in brackets. The top of the figure shows one chromosome, depicted as a giant tour. Dashed edges represent possible returns to the depot. The value on each edge is the traveling cost between two genes or between one gene and the depot. This cost is null between two genes denoting the same location, like b_1 and b_2. The auxiliary graph H is given in the middle, with the shortest path in boldface. Some arcs of H are not indicated, to avoid overloading the figure. The value close to each

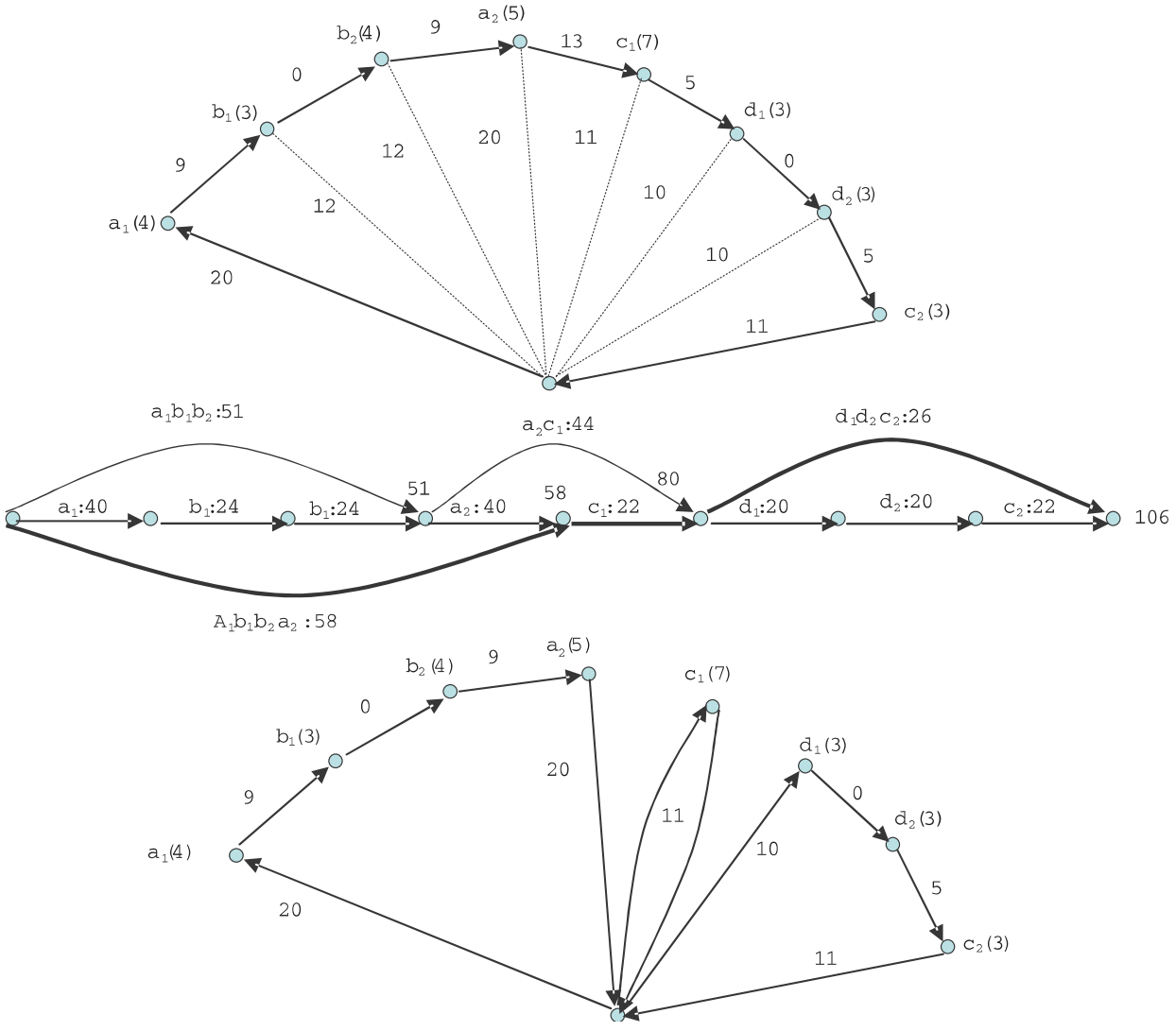

Fig. 3. Example for the MC-VRP splitting procedure.

node is the cost of a shortest path from node 0 to that node. The lower part of the figure gives the resulting MC-VRP solution: three routes with a total cost equal to 106.

Algorithm 2 gives a version based on Bellman's algorithm for directed acyclic graphs. The label V_j represents the cost of the shortest path from node 0 to node j in the auxiliary graph H, but H is considered implicitly, i.e., without being generated a priori. The *for* and *repeat* loops enumerate all feasible subsequences or routes $(S_i, \ldots, S_j)$. Each feasible route is evaluated and its cost is used to immediately update the label V_j of gene S_j. Each time a label V_j is improved, its predecessor on the path is memorized in P_j. The cost of the MC-VRP solution is given at the end by $V(\rho)$. If necessary, for instance to return the best solution at the end of the MA, the routes can be extracted by backtracking from $P(\rho)$. The complexity of this algorithm is proportional to the number of feasible subsequences, i.e., $O(\rho^2)$ in the worst case (recall that $\rho \leqslant nm$ is the number of non-null demands). Hopefully, the algorithm is much faster in practice because most long subsequences violate compartment capacities. Moreover, the space complexity is only $O(\rho)$, instead of $O(\rho^2)$ if H were explicitly generated.

Algorithm 2. Splitting algorithm with implicit auxiliary graph

set label V_0 to 0 and each other label V_i to $+\infty$, $1 \leqslant i \leqslant \rho$
for $i = 1$ to ρ **do**
  $cost = 0$
  set $load(p)$ to 0, $1 \leqslant p \leqslant m$
  $j = i$
  repeat
    $k = cust(S_j)$; $p = prod(S_j)$ (customer and product index for S_j)
    $load(p) = load(p) + q_k$
    if $i = j$ **then**
      $cost = c_{0,k} + c_{k,0}$
    else
      $cost = cost - c_{cust(S_{j-1}),0} + c_{cust(S_{j-1}),k} + c_{k,0}$
    end if
    if $(load(p) \leqslant Q_p)$ and $(cost \leqslant L)$ **then**
      if $V_{i-1} + cost \leqslant V_j$ **then**
        $V_j = V_{i-1} + cost$
        $P_j = i - 1$
      end if
      $j = j + 1$
    end if
  until $(j > \rho)$ or $(load(p) > Q_p)$
end for

The splitting procedure ends by making consecutive the genes which represent the same customer in each route. For example, given a route $T = (1, 3, 7, 6, 4)$ and $m = 2$ products, genes 3 and 4 represent the two products of customer 2 and they are not adjacent. As the triangle inequality holds and the distance between genes 3 and 4 is equal to zero, T is dominated by the route $T' = (1, 3, 4, 7, 6)$ obtained by moving gene 4 just after gene 3.

3.4. Initial population

The population Π is an array of σ chromosomes, each with two fields: a sequence S of customers and the solution cost computed by the splitting procedure. Two strategies S1 and S2 have been tested to initialize Π. S1 simply fills Π using random permutations of customers. S2 is more involved and requires some explanations. A constructive heuristic called *CH* is used to build $\sigma/2$ good and well-diversified solutions. The other $\sigma/2$ solutions are random, like in S1. In general, population-based metaheuristics are enhanced if they start from such solutions: this may accelerate convergence, decrease the risk of being trapped at a local optimum and favor a wider search of solution space.

The heuristic *CH* in S2 starts by dividing the given MC-VRP instance into m classical VRPs, one by product. The problems obtained are solved with the Clarke and Wright savings heuristic [10]. This gives m VRP solutions $E_1, \ldots, E_m$. By means of random swaps of customers in the routes (respecting vehicle capacity), a well-diversified list U_p of $\sigma/2$ VRP solutions is derived from each VRP solution E_p. The distance for R-permutations [11], also called *broken pairs distance*, is used for diversification: for two solutions A and B, it consists of counting the number of pairs of consecutive customers in A that are no longer consecutive in B.

Using the m lists U_p, each containing $\sigma/2$ VRP solutions, $\sigma/2$ MC-VRP solutions are assembled. Let U_{pk} denote the VRP solution of rank k in list U_p. The MC-VRP solution Π_k, $k = 1, 2, \ldots, \sigma/2$, is obtained by merging the m VRP solutions $U_{1k}, U_{2k}, \ldots, U_{mk}$: U_{1k} is merged with U_{2k}, the resulting solution with U_{3k}, and so on. Let A be the provisional MC-VRP solution at iteration p, i.e., A already contains $p - 1$ products. Merging A and U_{pk} consists of concatenating each route T of U_{pk} to the nearest route T' of A, using the distance measure (15). The customers of T are added to T' one by one. If the incumbent customer i is already in T', his demand q_{ip} is aggregated with the

P_1:	1	2	5	4	|	3	6	7	10	11	|	8	9	12
P_2:	5	6	11	12	|	1	2	9	3	4	|	7	8	10
C:	1	2	9	4	|	3	6	7	10	11	|	8	5	12

Fig. 4. Example of OX crossover.

amount delivered in T', otherwise a least-cost insertion of i in T' is performed.

$$d(T, T') = \sum_{x \in T} dist(x, T') \quad \text{where } dist(x, T') = \min_{y \in T'} c_{xy}. \tag{15}$$

Starting from an empty population, each new MC-VRP solution computed by *CH* is added to Π, provided the spacing rule (11) is satisfied. In practice, $\sigma/2$ solutions are obtained, exceptionally less because the well-diversified solutions easily pass the dispersal test. The remaining solutions of Π consist in random permutations of customers. The resulting population is finally ordered in ascending cost order. The best initial solution is then Π_1.

3.5. Selection and crossover

The lack of trip delimiters allows the use of a classical TSP crossover, the order crossover or OX [12]. It is applied at each MA iteration to two parents P_1 and P_2 selected by the *binary tournament method*: two solutions are randomly selected in Π and the best one is taken for P_1, the same process is repeated to get P_2. Two cutting points i and j such that $1 \leqslant i < j \leqslant \rho$ are randomly drawn. The subsequence $(P_1(i), \ldots, P_1(j))$ is copied into the elements $C(i)$ to $C(j)$ of the child-chromosome C. Then, P_2 is swept circularly from index $j+1$ onwards to complete C with the missed customers. Fig. 4 gives one example with $i = 5$ and $j = 9$.

Finally, the cost of the associated MC-VRP solution is evaluated with the splitting procedure. Recall that this procedure ends by regrouping in each route the genes that represent the same customer.

3.6. Local search

The local search is applied with a fixed probability p_{LS} to the offspring C, after each crossover. It operates on a complete solution with route delimiters (obtained by the splitting procedure) and not on the chromosome itself. This solution contains one node per gene (customer + product) but the splitting procedure has made adjacent in each route the nodes representing the products ordered by the same customer. A maximal sequence of such adjacent nodes in a route is called an *aggregate* in the sequel, even if it contains one single node.

Our local search is not allowed to split aggregates: this possibility is reserved to the crossover. Hence, each aggregate is considered as one single node during the local search. The number of aggregates μ in a solution ranges from $\mu = n$, when each customer appears in one aggregate only, to $\mu = nm$, when customers are visited once for each product but by distinct trips. Fortunately, the worst case may appear in initial random solutions but becomes quite unlikely after some MA iterations. The three kinds of moves below are evaluated.

1. *2-OPT* [13]. Let $succ(i)$ be the aggregate after aggregate i in its trip. The classical 2-OPT move consists of replacing two non-adjacent edges $(i, succ(i))$ and $(j, succ(j))$ by (i, j) and $(succ(i), succ(j))$ in a trip (the subsequence $(succ(i), j)$ is inverted). In our local search, 2-OPT moves are also applied to two edges $(i, succ(i))$ and $(j, succ(j))$ belonging to two distinct trips T and T'. Two cases must be evaluated: the two edges are either replaced by (i, j) and $(succ(i), succ(j))$ or $(i, succ(j))$ and $(j, succ(i))$. Compared to the single-trip case, compartment capacity violations must be checked for T and T'.
2. *Relocate* [13]. Remove one aggregate i from its current route to reinsert in another route.
3. *1-Interchange* [14]. Such moves exchange two aggregates pertaining to different routes.

The three kinds of moves can be enumerated in $O(\mu^2)$. For the inter-trip 2-OPT moves, this requires a data structure maintaining for each node in a route the cumulated load and cost since the beginning of that route. When a move makes consecutive in a trip two aggregates coming from two different trips, these two aggregates are merged.

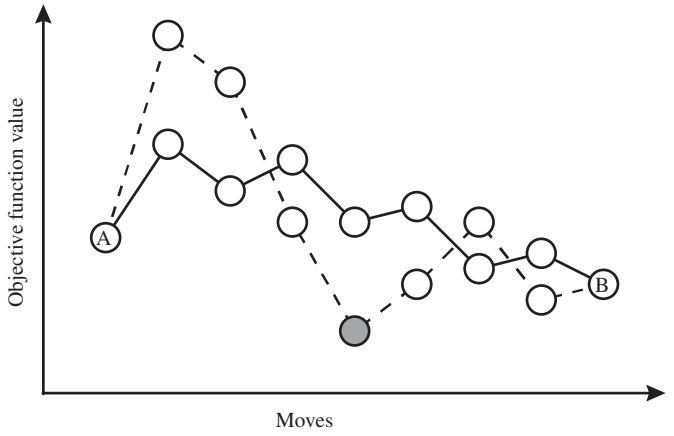

Fig. 5. The path relinking.

In each iteration of the local search, the neighborhoods defined by moves 1–2–3 are enumerated in this order. The enumeration is stopped as soon as one improving move is detected in the current neighborhood (the following neighborhoods are not inspected). The improving move is executed and the enumeration order is cyclically shift for the next iteration, i.e. the second iteration will check the moves in the order 2–3–1, the third one in the order 3–1–2 etc. The whole local search ends when no improving move can be found. The solution is then converted into a chromosome by concatenating its trips.

4. Path relinking

Path relinking was initially proposed to integrate intensification and diversification in tabu search [15]. It is based on the exploration of paths connecting pairs of high-quality solutions in the search space, with the hope to discover better solutions along these paths. This technique is in general used as a post-optimization procedure, at the end of a tabu search metaheuristic. Several versions are described in literature [16]. Usually, a small pool of elite solutions is collected during the execution of a metaheuristic. Then, one path is generated for each pair (A, B) of solutions. Such a path is a sequence of solutions in which A is transformed into B, by progressively introducing attributes of solution B. In general, a local search must be applied to new solutions, otherwise improvements may be marginal. Various strategies are also available to update the pool. For instance, the best new solution is added to the pool and a path from this new solution to any other pool solution is generated. This process stops when the best solution cannot be improved.

Fig. 5 shows a classical example [16], the path from A to B (solid line) yields no improved solutions, but the one from B to A (dotted line) produces three solutions better than A and B, with the best one in gray.

To the best of our knowledge, path relinking is applied here for the first time after a memetic algorithm. The pool for the MC-VRP is the population obtained at the end of the MA. This pool has not an excessive size because MAs with a diversification rule (like (11)) must use small populations, e.g., $\sigma = 20$, to avoid discarding too many child-solutions [8]. The path relinking directly operates on pairs (A, B) of chromosomes, with $\rho \leqslant nm$ genes.

To limit computation time, only n solutions are generated along a path, i.e., the path is a sequence $(S_0{=}A, S_1, \ldots, S_n{=}B)$. Solution S_k, $1 \leqslant k < n$, consists in the first $f = \lfloor \rho/n \rfloor k$ genes of B, completed by scanning A from left to right. When each customer asks for the m products, f is equal to mk. The local search is applied only if k is even, because the same local optimum is too often obtained when it is applied to two successive solutions of the path. This is also another way of limiting the computational effort.

The pool is updated as follows. The first iteration builds one path from A to B and one path from B to A (as described above), for each unordered pair $\{A, B\}$ of distinct solutions. If the best solution C on these two paths outperforms the best one found in the previous paths, S, then $S = C$. After exploring all paths, if S satisfies the spacing rule (11) and improves the worst solution in the pool or improves the current best solution, it replaces in the pool the worst one. If no

solution is replaced, the path relinking stops. Otherwise, the same process is repeated, except that the pairs examined must contain at least one new solution.

5. Tabu search

Tabu search (TS) is a meta-heuristic exploring the solutions space by moving at each iteration from the incumbent solution S to the best solution S' in its neighborhood $N(S)$, even if the objective function deteriorates. To avoid coming back to recently visited solutions, a tabu list acting as short-term memory of visited solutions must be implemented. A diversification mechanism, based for instance on long-term memory, is often added to access unexplored regions of the search space. Numerous applications like job shop scheduling, graph coloring, TSP and other vehicle routing have been successfully tackled by TS, see for instance [15] for an overview. The proposed TS starts from one initial feasible solution and improves it. This solution is built exactly like the initial solutions in the MA: the MC-VRP instance is divided into m classical VRPs, one per product, these VRPs are solved using the Clarke and Wright heuristic and the resulting m VRP solutions are finally assembled into one MC-VRP solution (see Section 3.4 for details). In the following the implemented TS is completely described by specifying the other components, namely: neighbourhood, short-term memory, diversification mechanism and stop condition.

5.1. Neighborhood

The neighborhood used by the proposed TS is similar to the one already presented in Section 3.6 for the MA. Nevertheless, instead of using a first-improvement strategy, the best non-tabu move in the neighborhood is selected to move from the incumbent solution S to a new solution S'. Moreover, the capacity and length constraints (see constraints (7) and (8) in the proposed formulation) are relaxed. The capacity and length violations are multiplied by two coefficients and the penalized term added to the objective function. The scope is to enlarge the search space by visiting infeasible solutions. The resulting new cost function $cost'(S)$ defined in (16) is inspired by the one proposed by Gendreau et al. for the VRP [17]:

$$cost'(S) = cost(S) + \delta Q(S) + \zeta D(S), \tag{16}$$

where $cost(S)$ is the routing cost of solution S (the only cost used in the MA), $Q(S)$ represents the sum of capacity violations for all routes and $D(S)$ indicates the total overtime. The penalty factors δ and ζ are self-adjusting parameters, initialized to given values $\bar{\delta}$ and $\bar{\zeta}$ and dynamically modified during the search. The penalty δ (ζ) is reduced by a constant τ after each block of ϕ_{inf} successive solutions respecting the capacity (duration) constraints and multiplied by τ after each block of ϕ_{inf} successive solutions violating the capacity (duration) constraint.

5.2. Short-term memory

The tabu list is the most used short-term memory to avoid looping back to already visited solutions. In its simplest form, a tabu list contains the solutions that have been visited in the recent past. Nevertheless, since recording a whole solution is space consuming, a better way of doing consists in memorizing a set of attributes able to characterize the solution. In the proposed neighborhood each move is completely identified with a set of arcs. More precisely, the short-term memory is implemented as a matrix L of $\rho \times \rho$ dimension. Each element $L(i, j)$ records the iteration in which the edge (i, j) has been removed from the solution. An arc removed at iteration t is forbidden to be reinserted in the solution until iteration $t + \theta$. The size of the tabu list θ takes its values in $[\theta_{\min}, \theta_{\max}]$ starting from θ_{init}. The parameter θ is updated according to the quality of the solutions obtained during the recent moves. After each improvement of the current objective function, θ is updated as $\theta = \max(\theta - 1, \theta_{\min})$, with the aim of intensifying the search around this solution. Otherwise, after ϕ_{LT} consecutive moves deteriorating the value of the visited solutions, the size of the tabu list is updated as $\theta = \min(\theta + 1, \theta_{\max})$. In the literature the aspiration criteria have been introduced for accepting solutions that improve the best one even if tabu. This feature has not been kept in the final version since it did not bring any significant improvement when evaluated during the preliminary testing.

5.3. Diversification

The intensification phase consists of focusing the exploration in the promising regions, while the diversification phase consists of guiding the search into different regions less explored. In the literature there are many ways for diversifying or intensifying the search of a TS method (see for example [15]). The simpler and straightforward way of doing it consists of increasing or decreasing the size of the tabu list, as already explained. The idea of passing through infeasible solutions, described in Section 5.1, can be seen as a diversification method as well.

The third diversification strategy applied consists of exploiting a special type of *restart* as already proposed in [18]. The idea is that the *TS* restarts the exploration of the solution space from a solution called $\bar{S}$ whenever it gets trapped in a local minimum. $\bar{S}$ represents the best feasible solution evaluated, but not visited, by the *TS* algorithm during its search. The guess is that $\bar{S}$ has good properties to be explored. To update $\bar{S}$ during each iteration t, the algorithm explores the neighborhood $N(S)$ of the incumbent solution S, and saves two solutions S' and T. S' represents the best non-tabu solution in $N(S)$ and it is used to continue the search process. Note that S' can be infeasible, since the solutions visited by the *TS* may violate capacity or length constraints. T is the best non-tabu *feasible* solution obtained in $N(S)\backslash\{S'\}$ and at each iteration t, $\bar{S}$ is updated as $\bar{S} = \text{argmin}(cost(\bar{S}), cost(T))$. After $\gamma_{\max}$ iterations without improving the best feasible solution found so far or after $\nu_{\max}$ iterations since the last restart, the search "jumps" from the current solution S to $\bar{S}$ and starts the search again with an empty tabu list. $\gamma_{\max}$ must be chosen carefully to avoid a premature restart. The maximal number of restarts is fixed as $\eta_{\max}$ but this process can be stopped if after $\chi_{\max}$ restarts the best solution is not improved.

The last diversification strategy reduces the time spent on the paths (composed of infeasible solutions) that link feasible solutions in the penalized search space. After ϕ_{viol} successive iterations violating the constraints, the route with the maximum violation value is divided into two routes. The new route reuses the last φ % customers of the old route. This splitting may separate nodes even if they represent the same customer (aggregate). Two routes with smaller length and capacity are obtained and this may reduce or even cancel the penalty element in (16). This technique is not mandatory but substantially accelerates the search.

5.4. Algorithm overview

The number of iterations in the *TS* algorithm varies between $\gamma_{\max} \times \chi_{\max}$ and $\nu_{\max} \times \eta_{\max}$. Algorithm 1 gives the general structure of the tabu search, with its restart process, while Algorithm 1 describes the procedure *Search* used between two restarts.

Algorithm 3. Structure of the whole *TS* algorithm with restarts

split the MC-VRP into m VRPs and solve them using Clarke and Wright method
build an initial feasible solutions S using the VRP solutions (see Section 3.4)
$\bar{S} = S^* = S$
$\eta = 0$ {Number of restarts}
$\chi_{impr} = 0$ {Number of restarts without improvement}
initialize the penalization parameters
initialize the tabu list
repeat
  $\eta = \eta + 1$
  Search($\bar{S}$, S, *improved*)
  if *improved* **then**
    $\chi_{impr} = 0$
  else
    $\chi_{impr} = \chi_{impr} + 1$
  end if
  $S = \bar{S}$
until $(\eta = \eta_{\max})$ or $(\chi_{impr} = \chi_{\max})$

Algorithm 4. Procedure *Search*($\bar{S}$, *S*,*improved*).

$\theta = \theta_{\text{init}}$
$\gamma = 0$ {Number of iterations without improvement}
$v = 0$ {Total number of iterations}
$\chi = 0$ {Number of consecutive infeasible solutions}
noiwis $= 0$ {Number of iterations without improving the incumbent solution}
cifd=*cfd*=*cfq*=*cifq*=0 {Number of consecutive feasible and non feasible solutions (capacity and length)}
repeat
  $v = v + 1$
  S' = the best non-tabu solution in $N(S)$
  T = the best feasible non-tabu solution in $N(S)\backslash\{S'\}$
  $\bar{S} = \text{argmin}(cost(\bar{S}), cost(T))$
  if $cost(S') < cost(S)$ **then**
    $\theta = \max(\theta - 1, \theta_{\min})$
    noiwis $= 0$
  else
    noiwis $=$ *noiwis* $+ 1$
  end if
  if *noiwis* $= \phi_{\text{LT}}$ **then**
    $\theta = \min(\theta + 1, \theta_{\max})$
    noiwis $= 0$
  end if
  if $cost(S') < cost(S^*)$ and S' is feasible **then**
    $S^* = S'$; $\gamma = 0$; *improved* $=$ *TRUE*;
  else
    $\gamma = \gamma + 1$
  end if
  $S = S'$
  store the attributes of S in the tabu list
  if S infeasible **then**
    $\chi = \chi + 1$
  else
    $\chi = 0$
  end if
  if $\chi = \phi_{\text{inf}}$ **then**
    split the route with maximal violation into two routes
    $\chi = 0$
  end if
  test the solution and update *cifd*, *cfd*, *cfq*, *cifq* accordingly
  check *cifd*, *cfd*, *cfq*, *cifq* and update ζ and δ accordingly (see 5.2)
until $(\gamma = \gamma_{\max})$ or $(v = v_{\max})$

6. Computational results

The algorithms are coded in C and have been tested on a PC Pentium 4 at 2.4 GHz. To the best of our knowledge, no instances are publicly available for the MC-VRP, which is a new problem. This is why the following section describes how instances have been created. The parameters chosen are given in Section 6.2 and the results obtained reported in Section 6.3.

6.1. Instances used and purpose

Twenty well-known instances for the VRP have been transformed into two sets of MC-VRP instances with $m = 2$, by creating two compartments and splitting the request of each customer into two amounts. The first 14 problems (*vrpnc*)

Table 1
Parameters

MA parameters	*TS* parameters
$\sigma = 20$	$\tau = 1.3$
$\Delta = 0.5$	$\phi_{\text{inf}} = \phi_{\text{LT}} = \phi_{\text{viol}} = 10$
$\alpha_{\max} = 30000$	$\gamma_{\max} = 500$
$\beta_{\max} = 10000$	$\nu_{\max} = 2000$
$p_{\text{LS}} = 0.5$	$\eta_{\max} = 50$
	$\varphi = 30\%$
	$\bar{\zeta} = 1$
	$\bar{\delta} = 1$
	$\theta_{\text{init}} = 7$
	$\theta_{\min} = 5$
	$\theta_{\max} = 15$
	$\chi_{\max} = 5$

Table 2
Results of *MA* on a population initialized by S1 or S2

File	n	Strategy S1		Strategy S2	
		Cost	Time	Cost	Time
vrpnc1	50	524.61	22.94	524.61	23.66
vrpnc2	75	848.27	57.59	**842.68**	55.08
vrpnc3	100	**835.33**	94.22	853.23	62.33
vrpnc4	150	**1068.34**	180.78	1070.90	207.08
vrpnc5	199	1341.08	490.16	**1330.39**	410.30
vrpnc6	50	559.32	9.98	559.32	19.45
vrpnc7	75	**924.08**	23.2	928.07	45.94
vrpnc8	100	**879.1**	70.52	892.17	57.02
vrpnc9	150	**1172.29**	125.83	1199.86	157.08
vrpnc10	199	**1456.79**	497.03	1461.20	500.81
vrpnc11	120	1162.63	134.25	**1044.65**	126.52
vrpnc12	100	819.56	95.83	819.56	76.66
vrpnc13	120	1569.8	145.88	**1541.23**	139.55
vrpnc14	100	**866.37**	42.64	866.79	55.47
Average		**1001.96**	**142.20**	**995.33**	**138.36**

are due to Christofides and contain 50–199 customers. Instances 6–10, 13 and 14 include a route length restriction. The last six instances come from Eilon: their names begin by "E", followed by the number of customers (76–484). All these instances may be downloaded from Beasley's *OR Library* [19].

Although our implementation handles more compartments, we took $m = 2$ for the testing because: (a) the problem is already difficult enough with 2, (b) it corresponds to the widespread case of frozen food distribution, (c) it enables an easier comparison with VRP solutions, and (d) the demands obtained by dividing the original demands in 3 or more would be too small.

The first set of instances has been obtained dividing the capacity of each vehicle and each customer request into two equal parts. Therefore, each solution for the original VRP is feasible for the MC-VRP.

The second set has been obtained dividing randomly each customer request into two parts. For each customer i, the request for the first product is $q_{i1} = q_i/k$, where k is a random integer in [3,5] and q_i is the demand in the VRP. The request for the second product is $q_{i2} = q_i - q_{i1}$. All resulting requests are real numbers, i.e., they are not rounded to the closest integer. The capacity Q_1 of the first compartment is determined as follows from the average demand $\overline{Q}_1$ for the first product, the average demand $\overline{Q}_2$ for the second product, and the vehicle capacity Q in the original VRP: $Q_1 = (Q \times \overline{Q}_1)/(\overline{Q}_1 + \overline{Q}_2)$. The capacity of the second compartment is given by $Q_2 = (Q \times \overline{Q}_2)/(\overline{Q}_1 + \overline{Q}_2)$.

Table 3
Improvement of CH solution for the true MC-VRP (in %)

	With MA	With TS
Set 1	5.6	2.0
Set 2	33.4	34.0

Table 4
Results of *MA* and *TS* using the first set of instances—standard setting of parameters

File	n	VRP	MA						TS						Δ%
			MC-VRP-WS		MC-VRP				MC-VRP-WS		MC-VRP				
			Cost	Time	Cost	Time	Stops	Gain %	Cost	Time	Cost	Time	Stops	Gain %	
vrpnc1	50	524.6	524.6	17.4	524.6	23.7	50	0.0	524.6	19.6	524.6	15.8	50	0.0	0.0
vrpnc2	75	835.3	855.8	25.4	**842.7**	55.1	75	1.6	850.0	56.3	851.8	21.5	75	−0.2	−1.1
vrpnc3	100	826.1	876.8	21.8	853.2	62.3	100	2.8	831.3	50.9	**835.2**	95.3	100	−0.5	2.2
vrpnc4	150	1028.4	1089.6	93.9	1070.9	207.1	150	1.8	1061.1	285.4	**1055.1**	411.2	151	0.6	1.5
vrpnc5	199	1291.4	1389.6	115.9	**1330.3**	410.3	199	4.5	1348.3	403.6	1348.8	809.2	199	0.0	−1.4
vrpnc6	50	555.4	571.4	6.1	**559.3**	19.4	50	2.2	575.9	12.3	560.0	14.2	50	2.8	−0.1
vrpnc7	75	909.7	933.0	39.2	**928.1**	45.9	77	0.5	970.8	21.8	937.1	54.1	77	3.6	−1.0
vrpnc8	100	865.9	969.2	18.7	**892.2**	57.0	101	8.6	888.6	60.4	894.5	105.2	100	−0.7	−0.2
vrpnc9	150	1162.5	1230.9	98.7	1199.86	157.1	150	2.6	1232.1	98.3	**1184.2**	395.6	152	4.0	1.3
vrpnc10	199	1395.8	1520.1	140.2	**1461.2**	500.8	199	4.0	1538.6	35.2	1467.6	395.5	199	4.8	−0.4
vrpnc11	120	1042.1	1046.1	47.8	1044.65	126.5	122	0.1	1043.3	142.5	**1043.8**	64.9	120	0.0	0.1
vrpnc12	100	819.6	820.6	18.2	**819.6**	76.7	101	0.1	819.5	63.1	822.0	34.9	100	−0.3	−0.3
vrpnc13	120	1541.1	1547.4	76.4	**1541.2**	139.5	121	0.4	1582.2	44.3	1542.9	167.7	120	2.5	−0.1
vrpnc14	100	866.4	866.9	23.3	**866.8**	55.5	100	0.0	868.6	43.5	867.1	42.1	100	0.2	−0.0
E072-04f	72	241.9	241.9	11.7	244.8	11.4	72	−1.2	244.5	26.7	**241.9**	28.0	72	1.1	1.2
E076-08s	76	742.6	761.9	15.3	749.5	32.6	78	1.7	748.5	24.7	**747.9**	22.6	76	0.1	0.2
E076-07u	76	690.8	703.5	15.1	690.8	27.9	77	1.9	692.2	24.9	**690.2**	29.5	76	0.3	0.0
E135-07f	135	1162.9	1193.2	47.3	**1165.6**	160.2	137	2.4	1174.4	95.9	1179.21	140.7	135	−0.4	−1.1
E241-22k	241	666.8	748.6	504.4	743.8	306.4	241	0.6	741.4	386.9	**741.2**	805.8	241	0.0	0.3
E484-19k	484	1137.2	1250.1	1643.6	1204.1	2055.9	485	3.8	1149.5	3591.7	**1148.9**	6459.7	484	0.1	4.8
Average		**915.3**	**957.1**	**149.1**	**936.6**	**227.5**		**1.9**	**944.3**	**272.4**	**934.2**	**505.7**		**0.9**	**0.3**

The proposed testing consists of comparing (a) the MA with the TS and (b) the MC-VRP with what we call the *MC-VRP-WS* (MC-VRP without splitting), a kind of VRP with two capacities and two products, but in which it is not allowed to split the set of products ordered by a customer. To handle this MC-VRP-WS, we simply modified the MA and the TS to get two versions in which all splitting instructions are inhibited. Although our algorithms are not designed for the classical VRP, the purpose of the first set of instances is also to check whether we are not too far from best-known VRP solutions.

6.2. Parameters

After some experimentations, the parameters have been set to the values reported in Table 1. For the *MA*, σ represents the size of population. The threshold used as diversity criteria in Eq. (11) is Δ. Local search is applied with probability p_{LS}. The number of iterations of the *MA* without improving the best solution is β_{max}, while α_{max} is the maximum number of iterations. The *TS* restarts η_{max} times, and χ_{max} limits the number of runs without improving the best solution. The maximum number of moves without improving the best feasible solution in each restart is γ_{max}. ν_{max} is the number of iterations of each restart. ϕ_{inf}, ϕ_{LT}, ϕ_{viol} have been set to the same value and respectively count the number of solutions visited before updating θ, δ or ζ, or splitting the route with the greater penalty violation. $\bar{\zeta}$ and $\bar{\delta}$ are the initial values for the penalized parameters used in Eq. (16). The size of the tabu list is updated in the interval $[\theta_{min}, \theta_{max}]$, starting from θ_{init}.

Table 5
Results of *MA* and *TS* using the second set of instances—standard setting of parameters

File	n	MA						TS						Δ%
		MC-VRP-WS		MC-VRP				MC-VRP-WS		MC-VRP				
		Cost	Time	Cost	Time	Stops	Gain %	Cost	Time	Cost	Time	Stops	Gain %	
vrpnc1	50	558.8	17.4	**548.36**	22.5	51	1.9	556.1	15.3	550.54	12.75	50	1.0	−0.39
vrpnc2	75	888.6	25.5	874.49	45.1	75	1.6	863.6	13.9	**873.59**	56.00	75	−1.1	0.1
vrpnc3	100	878.4	21.8	843.39	91.9	100	4.1	837.6	39.8	**832.89**	158.97	100	0.6	1.26
vrpnc4	150	1089.1	93.9	1078.14	158.8	150	1.0	1070.7	109.7	**1075.99**	346.98	151	−0.5	0.19
vrpnc5	199	1408.5	115.9	1368.82	487.0	199	2.9	1361.4	208.4	**1362.75**	390.78	199	−0.1	0.44
vrpnc6	50	569.4	16.5	558.68	27.1	50	1.9	563.4	10.2	**558.60**	29.89	50	0.8	0.01
vrpnc7	75	955.1	39.2	959.85	18.1	75	−0.5	949.0	22.0	**952.59**	33.56	77	−0.4	0.76
vrpnc8	100	958.9	18.7	**890.08**	101.3	101	7.7	916.2	18.3	894.36	113.13	100	2.4	−0.47
vrpnc9	150	1262.7	98.7	1224.87	154.2	150	3.1	1290.8	8.6	**1186.15**	310.56	151	8.8	3.26
vrpnc10	199	1509.1	140.2	**1475.80**	412.5	199	2.3	1490.2	190.3	1500.26	368.14	201	−0.7	−1.63
vrpnc11	120	1122.9	47.8	**1113.38**	94.7	122	0.9	1201.6	27.9	1155.19	79.41	121	4.0	−3.61
vrpnc12	100	926.5	18.2	**906.94**	33.7	101	2.2	934.1	15.8	907.29	91.33	100	3.0	−0.03
vrpnc13	120	1542.4	76.4	**1541.23**	141.3	121	0.1	1582.3	21.9	1542.99	164.28	122	2.5	−0.11
vrpnc14	100	966.5	23.3	**934.73**	61.7	101	3.4	1141.6	35.7	941.36	53.27	100	21.3	−0.7
E072-04f	72	263.6	11.7	**262.45**	12.4	72	0.4	262.3	5.6	262.74	13.58	72	−0.1	−0.11
E076-08s	76	793.5	15.4	756.68	42.7	77	4.9	772.2	13.9	**748.36**	52.41	76	3.2	1.11
E076−07u	76	702.2	15.1	705.79	50.7	76	−0.5	697.8	16.5	**699.20**	21.36	76	−0.2	0.94
E135-07f	135	1233.2	47.3	1256.80	71.5	137	−1.9	1235.2	51.9	**1248.30**	260.58	135	−1.0	0.68
E241-22k	241	796.7	504.5	792.68	588.3	241	0.5	787.8	202.9	**771.21**	919.26	241	2.1	2.78
E484-19k	484	1240.9	1643.6	1224.02	1545.9	485	1.4	1177.3	2122.5	**1175.47**	3878.50	484	0.2	4.13
Average		**983.3**	**149.4**	**965.8**	**209.84**		**1.9**	**984.5**	**157.6**	**961.9**	**367.7**		**2.3**	**0.4**

6.3. Results

The obtained results are reported in Tables 2–5. All these tables have several columns in common: the ones with headers *File*, *n*, *Cost* and *Time* respectively indicate the instance name, the number of customers, the solution value obtained and the corresponding running time in seconds. In Tables 4 and 5, the *MC-VRP-WS* columns give solution values obtained by *MA* or *TS* when the orders of customers cannot be split, while the *MC-VRP* columns list the results of the true MC-VRP. The column *Stops* show the total number of stops in a solution: a value equal to n indicates a solution without splitting. The column *Gain* gives for *MA* and *TS* the gain in % when splitting is permitted. Finally, the column Δ reports the gain % when *TS* is used instead of *MA*. In all cases, the gain of a method B over a method A is given by $(A - B)/B \times 100$. The *VRP* column used only in Table 4 reports the best-known solutions for the VRP, as listed in [20].

Table 2 presents the results of a test with the *MA* on the 14 instances of the first set (the ones derived from *vrpnc* files). The algorithm is executed either from an initial population of random solutions (strategy S1) or containing 50% of good solutions computed with the heuristic described in 3.4 (strategy S2). The table shows that there is no clear enough computational evidence for choosing one strategy instead of the other, indicating that the *MA* is rather robust with regard to the quality of the initial population. However, S2 has been used in the other tests, because of its slightly better average solution values.

The small Table 3 concerns the true MC-VRP. It provides the savings brought by *MA* or *TS* to the solution of the constructive heuristic *CH*. The merging process in *CH* seems efficient for the instances of the first set: it offers little margin for improvement to the metaheuristics.

Table 4 provides a comparison between *MA* and *TS* and between each algorithm and its MC-VRP-WS version, on the first set of instances. Costs in boldface indicate which algorithm is the best for the true MC-VRP. On average, splitting improves the results by 1.9% for *MA* and 0.9% for *TS*. For the MC-VRP without splitting, *TS* is slightly better than *MA*, with an average saving of 0.3% (column Δ). Nevertheless, *MA* is twice faster than *TS* on average. The largest discrepancy is achieved on the largest instance *E484-19k*: *TS* requires around two hours, versus half an hour for *MA*. Without this large instance, the average running times are much smaller: 2.1 min for *MA* and 3 min for *TS*.

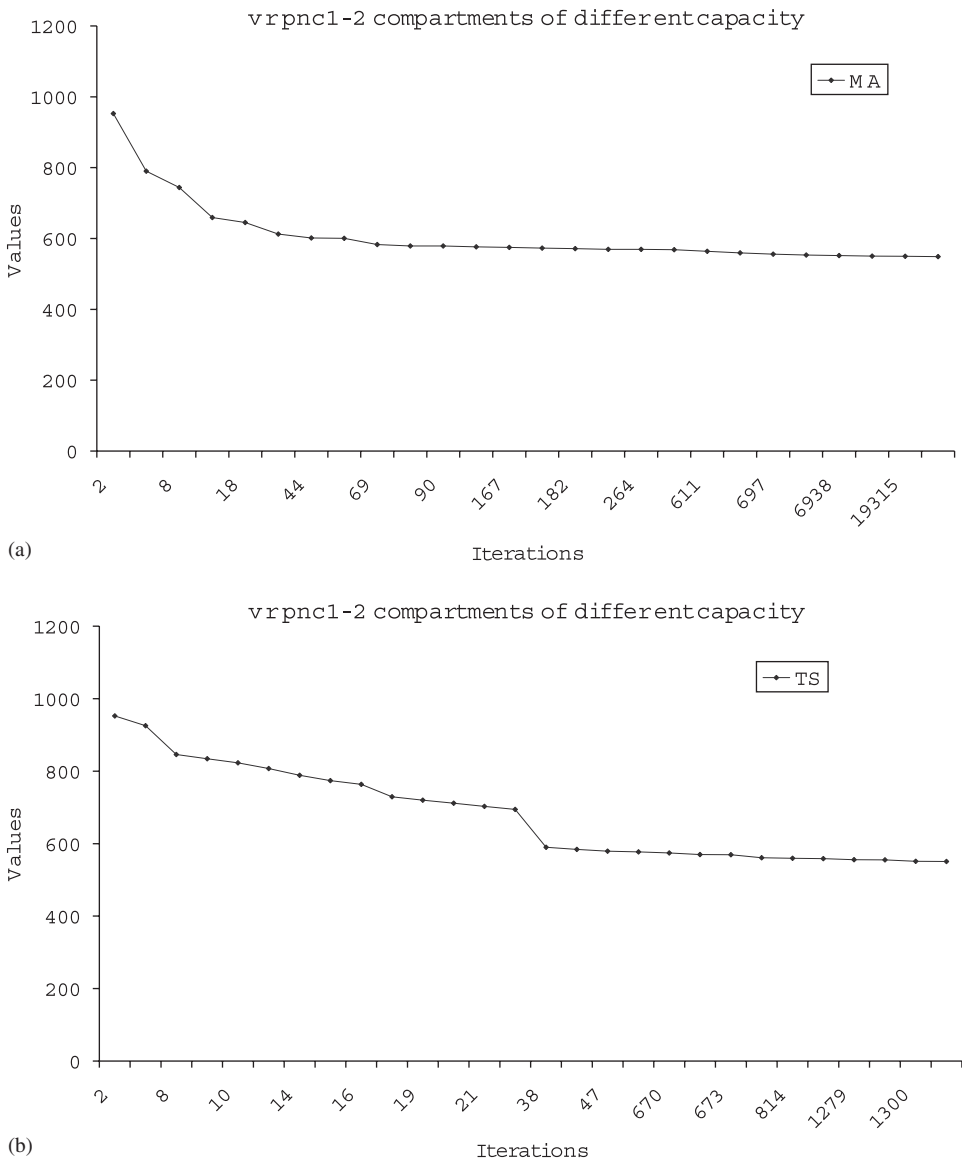

Fig. 6. Convergence of *MA* and *TS* on instance vrpnc1.

The interest of the first set of instances is also to compare with the VRP. The average deviation of our best method (*TS*) to the best-known VRP solution is only 1.02%. This is quite remarkable because our algorithms are not specially designed for the VRP and the best-known VRP solutions reported in literature were obtained by using various metaheuristics and settings of parameters. Moreover, the VRP solution is hard to find because of its special structure in a MC-VRP context: the two products for each customer must be regrouped in the same trip. Since our main goal is to solve the general MC-VRP, we have used for both sets of instances the setting of parameters which gives the best solutions on the second set. Therefore, the deviation could be reduced by using a better suited setting for the first set.

Table 5 provides the same comparisons as in Table 4, but this time for the second set of instances. On these more realistic problems, the saving brought by splitting is more important: 1.9% for *MA* and 2.3% for *TS*, even if each algorithm does not use splitting for 8 out of 20 solutions. Again, *TS* is only slightly better than *MA* (saving of 0.4%), at the expense of a doubled running time. It is also interesting to note that *MA* behaves better on instances 11 to 14, which are the only clustered ones.

Fig. 6 illustrates the convergence of *MA* (left curve) and *TS* (right curve) on instance *vrpnc1*. Iterations correspond to the number of crossovers for the *MA*, and to the number of neighborhoods visited for the *TS*. At the beginning, *MA* converges faster than *TS*, but *TS* shows two strong improvements which could be explained by its more sophisticated diversification mechanisms.

7. Conclusions

This paper introduces the MC-VRP, a problem not yet studied in literature in spite of its important practical applications. Three algorithms have been proposed to solve it: a constructed heuristic, a memetic algorithm combined with a path relinking method used as post optimization, and a tabu search. These methods have been tested using two sets of problems, obtained by adapting 20 known instances for the classical VRP. As no published method is available for comparison, each algorithm is compared with a version in which it is not allowed to split the set of products ordered by a customer. On both sets, splitting improves the results on average. In general, *TS* provides slightly better solutions than *MA* but requires more computational time. Moreover, for the first set, the results of our metaheuristics are very close to the best-known VRP solution values.

References

[1] Toth P, Vigo D. The vehicle routing problem. SIAM monographs on discrete mathematics and application. Philadelphia, PA: SIAM; 2002.
[2] Dror M, Trudeau P. Saving by split delivery routing. Transportation Science 1989;23:141–5.
[3] Avella P, Boccia M, Sforza A. Solving a fuel delivery problem by heuristic and exact approaches. European Journal of Operational Research 2004;152:170–9.
[4] Brown GG, Graves WG. Real-time dispatch of petroleum tank trucks. Management Science 1981;27:19–31.
[5] Brown GG, Ellis CJ, Graves WG, Ronen D. Real-time, wide area dispatch of mobil tank trucks. Interfaces 1987;17:107–20.
[6] Van der Brugen L, Gruson R, Salomon M. Reconsidering the distribution of gasoline products for a large oil company. European Journal of Operation Research 1995;81:460–73.
[7] Moscato P. Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F, editors. New ideas in optimization. New York: McGraw-Hill; 1999. p. 219–34.
[8] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 2004;31(12): 1985–2002.
[9] Beasley JE. Route-first cluster-second methods for vehicle routing. Omega 1983;11:403–8.
[10] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964;12:568–81.
[11] Campos V, Laguna M, Marti R. Context-independent scatter search and tabu search for permutation problems. INFORMS Journal on Computing 2005;17:111–22.
[12] Oliver IM, Smith DJ, Holland JRC. A study of permutation crossover operators on the traveling salesman problem. In: Grefenstette JJ, editor. Proceedings of the second international conference on genetic algorithms. Hillsdale, NJ: Lawrence Erlbaum; 1987. p. 224–30.
[13] Croes GA. A method for solving traveling salesman problems. Operations Research 1958;6:791–812.
[14] Osman IH. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 1993;41:421–51.
[15] Glover F, Laguna M. Tabu search. Boston: Kluwer Academic Publishers; 1997.
[16] Glover F, Laguna M, Marti R. Fundamentals of scatter search and path relinking. Control and Cybernetics 2000;39(3):653–84.
[17] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[18] Dell'Amico M, Lodi A, Maffioli F. Solution of the cumulative assignment problem with a well-structured tabu search method. Journal of Heuristics 1999;5(2):123–43.
[19] ⟨http://people.brunel.ac.uk/∼mastjjb/jeb/info.html⟩.
[20] Laporte G, Gendreau M, Potvin JY, Semet F. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 2000;7:285–300.