Engineering Applications of Artificial Intelligence 22 (2009) 916–928

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Two memetic algorithms for heterogeneous fleet vehicle routing problems

Christian Prins

Institut Charles Delaunay (FRE CNRS 2848), Université de Technologie de Troyes BP 2060, 12 rue Marie Curie, 10010 Troyes Cedex, France

ARTICLE INFO

Article history:
Received 7 May 2008
Received in revised form
22 August 2008
Accepted 3 October 2008
Available online 3 December 2008

Keywords:
Vehicle routing
Heterogeneous fleet
Fleet mix problem
Memetic algorithm
Distribution
Metaheuristic

ABSTRACT

The vehicle routing problem (VRP) plays an important role in the distribution step of supply chains. From a depot with identical vehicles of limited capacity, it consists in determining a set of vehicle trips of minimum total length, to satisfy the demands of a set of customers. In general, the number of vehicles used is a decision variable. The heterogeneous fleet VRP (HFVRP or HVRP) is a natural generalization with several vehicle types, each type being defined by a capacity, a fixed cost, a cost per distance unit and a number of vehicles available. The vehicle fleet mix problem (VFMP) is a variant with an unlimited number of vehicles per type. This paper presents two memetic algorithms (genetic algorithms hybridized with a local search) able to solve both the VFMP and the HVRP. They are based on chromosomes encoded as giant tours, without trip delimiters, and on an optimal evaluation procedure which splits these tours into feasible trips and assigns vehicles to them. The second algorithm uses a distance measure in solution space to diversify the search. Numerical tests on standard VFMP and HFVRP instances show that the two methods, especially the one with distance measure, compete with published metaheuristics and improve several best-known solutions.

1. Introduction

The vehicle routing problem (VRP) plays an important role in supply chains, where they often arise in the first transportation step (to collect agricultural products, for instance) or in the final distribution phase toward customers. While the *travelling salesman problem* (TSP) is perhaps the most famous single-vehicle problem, the VRP is the standard academic problem with several vehicles.

The VRP can be defined on an undirected complete graph with one depot (node 0) and n customers indexed from 1 to n. A fleet of identical vehicles with limited capacity Q is based at the depot. Each customer has a known demand q_i and a distance d_{ij} is associated with each edge $[i, j]$. The goal is to design a set of trips of minimum total length to supply all customers. A trip is a cycle performed by one vehicle, starting and ending at the depot and visiting a subset of customers. Its total load must not exceed the vehicle capacity. Each customer must be visited by one single trip, i.e., split deliveries are not allowed. In general, the number of trips or vehicles used is not imposed, it is a decision variable.

The VRP is NP-hard and the current best exact methods are still limited to 100 customers (Fukasawa et al., 2006), explaining why larger instances require heuristics like constructive methods (Laporte and Semet, 2002) or metaheuristics (Cordeau et al., 2005). Recent surveys on the VRP can be found in Toth and Vigo (2002) and Cordeau et al. (2007).

The VRP is not very realistic, since most companies have heterogeneous fleets of vehicles. In the *vehicle fleet mix problem* (VFMP), the fleet is composed of t vehicle types. Each type k is defined by a capacity Q_k, a fixed cost f_k and a cost per distance unit v_k, often called variable cost. A trip of length L done by a vehicle of type k has a cost f_k+Lv_k. The goal is to compute a set of trips and to assign vehicles to trips to minimize total cost. Like in the VRP, the number of vehicles of each type is not limited. The VFMP appears in situations where the fleet is not yet purchased: it combines tactical decisions (selecting the number of vehicles to be acquired) and operational ones (computing the trips and the vehicles assigned to them).

The *heterogeneous fleet VRP* (HFVRP or HVRP) is a further generalization with a limited availability a_k for each vehicle type k. The VFMP can be viewed as a particular HVRP with $a_k = n$, since there are at most n trips (with one customer each). The VFMP and HVRP are obviously NP-hard, since they include the VRP as a special case.

This paper is structured as follows. Section 2 clarifies the intricate history of VFMP and HVRP and provides bibliographic references. Components for memetic algorithms (MAs) able to tackle both the VFMP and the HVRP are described in Section 3. Section 4 proposes two different MAs built from these basic bricks. In Section 5, these metaheuristics are compared with published methods on standard instances from literature.

E-mail address: christian.prins@utt.fr

0952-1976/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2008.10.006

Table 1
Published methods using instances from Golden et al.

Authors	Year	Kind	VFMP-F	VFMP-V	VFMP-FV	HVRP
Golden et al.	1984	Heuristics	Yes: 1–20	No	No	No
Taillard	1999	HCG	Yes: 13–20	Yes: 13–20	No	Yes: 13–20
Gendreau et al.	1999	TS	Yes: 3–6, 13–20	Yes: 13–20	No	No
Wassan and Osman	2002	TS	Yes: 1–20	Yes: 13–20	No	No
Renaud and Boctor	2002	Heuristics	Yes: 1–20	No	No	No
Lima et al.	2004	MA	Yes: 1–20	No	No	No
Tarantilis	2004	TA	No	No	No	Yes: 13–20
Choi and Tcha	2007	B&B	Yes: 3–6, 13–20	Yes: 3–6, 13–20	Yes: 3–6, 13–20	No
Li et al.	2007	RTR	No	No	No	Yes: 13–20

2. History of routing problems with heterogeneous fleets

Contrary to the VRP, the literature on VFMP and HVRP is still scarce. It is also confusing, because many papers mentioning the HVRP deal in fact with the VFMP and use fixed costs or variable costs, but not both. This section tries to clarify history by considering three VFMP versions: with fixed costs only (VFMP-F: all $f_k \neq 0$ and all $v_k = 1$), with variable costs only (VFMP-V: all $f_k = 0$ but distinct v_k values) and with both costs (VFMP-FV).

If fixed and variable costs grow with vehicle capacity, the fleet is said to be *correlated*. Uncorrelated fleets are not rare: consider a vehicle type 1 with $f_1 = 10$ and $v_1 = 1$ and a more recent type 2 consuming less gasoline, with $f_2 = 15$ and $v_2 = 5$: type 2 becomes cheaper beyond 10 km. It is important to see that methods for correlated fleets are not always adaptable to the uncorrelated case. For instance, if the fleet is correlated, an improving exchange of two customers in a route is detected by a negative total length variation. The detection is more involved if the fleet is uncorrelated and if the vehicle may be changed.

The VFMP with fixed costs (VFMP-F) was introduced by Golden et al. (1984). These authors described greedy heuristics and proposed 20 instances with 12–100 customers and 3–6 vehicle types, reused by all other authors. Instances 1–12 have 12 or 30 customers. Instances 13–20 are larger, with 50–100 customers. Instances 1, 2 and 7–12 are defined by non-Euclidean distance matrices, without node coordinates. In all other instances, node coordinates are provided and the travel cost between two nodes corresponds to the Euclidean distance. Since many heuristics require node coordinates, to define clusters for instance, most authors reuse Euclidean instances 3–6 and 13–20 and discard the others.

Osman and Salhi (1996) proposed the first metaheuristic for the VFMP-F, a tabu search (TS), and tested it on the 20 instances of Golden. Taillard (1999) developed a heuristic column generation (HCG) method: a tabu search requiring node coordinates is used to generate a large set of promising routes and a solution is obtained by solving a set partitioning problem whose columns correspond to these routes. Taillard discarded the smallest Euclidean instances (3–6), for which his tabu search produces too few columns. He also replaced fixed costs by variable costs in the other Euclidean VFMP-F files (13–20) to get eight new VFMP-V files. Finally, he built eight HVRP files by adding limited availabilities a_k to these VFMP-V files. His paper provides results for the three kinds of problems.

Two other tabu search heuristics were analyzed by Gendreau et al. (1999) and Wassan and Osman (2002). Both papers report results for the VFMP-F instances of Golden (only instances 3–6 and 13–20 for Gendreau et al.) and for the 8 VFMP-V instances of Taillard.

Renaud and Boctor (2002) elaborated constructive heuristics and improvement procedures for the VFMP-F, using the 20 Golden instances. Prins (2002) considered a special HVRP: vehicle types have distinct capacities and limited availabilities, but fixed and variable costs are absent: the objective function is the total length of the trips. He proposed constructive heuristics and a tabu search which can also tackle the multitrip case, i.e., each vehicle may perform several trips subject to a maximum range. The algorithms were tested on randomly generated instances and on a large real case with 775 customers.

Tarantilis et al. (2003) designed a threshold accepting algorithm (TA) for the HVRP and provided an improved version in 2004. They used the eight HVRP instances from Taillard. Lima et al. (2004) published a MA for the VFMP-F. Compared to the TS procedures of Gendreau et al. (1999) and Wassan and Osman (2002), this algorithm improves some best-known solutions to Golden's instances, at the expense of important running times on large instances (up to 49 min for 100 customers on a 2 GHz PC).

Yaman (2006) described lower bounds (LBs) for the VFMP-F. Choi and Tcha (2007) proposed a better bound based on column generation and used in a branch-and-bound (B&B) algorithm for the Euclidean VFMP. They added variable costs to Euclidean instances 3–6 from Golden and gave results for the VFMP-F and VFMP-V. They also tackled for the first time fixed and variable cost, giving 12 VFMP-FV problems. The B&B never finds a proven optimum but achieves small deviations to the LB, between 0.2% and 4%.

Li et al. (2007) published a record-to-record (RTR) travel metaheuristic for the HVRP. It outperforms the threshold accepting method of Tarantilis et al. (2004) on the eight HVRP instances from Taillard.

Table 1 summarizes this complicated history and shows that no author has tackled the four problems. Only Taillard (1999); Tarantilis et al. (2003, 2004) and Li et al. (2007) have studied a true HVRP, with a limited fleet. Only Golden et al. (1984); Renaud and Boctor (2002); Lima et al. (2004) and Choi and Tcha (2007) clearly indicate that a correlated fleet is assumed. In all other papers, all instances used have correlated fleets and there is no proof that the proposed solution methods could tackle uncorrelated fleets. Hence, our goal is to design effective MAs able to tackle the four problems, even when the fleet of vehicles is uncorrelated.

3. Components for MAs

A MA (Moscato's MA model, 1999), also called hybrid GA or genetic local search, is a genetic algorithm in which new solutions obtained by crossover and mutation are improved using a local search. This section describes components which are combined in Section 4 to give two MAs.

3.1. Chromosomes and basic evaluation

Each chromosome C is a permutation of customers, with implicit shortest paths linking consecutive nodes, but without trip delimiters. It can be viewed as a giant tour (TSP tour) for a vehicle with infinite capacity. It is evaluated using a tour splitting procedure *Split* which optimally partitions C into feasible routes, subject to the sequence. Using this strategy, the MAs search the set of TSP tours, which is much smaller than the set of VFMP or HVRP tours. Moreover, there is no loss of information, since we can derive the best possible VFMP or HVRP tour from each TSP tour.

We recall first the principle of *Split* for the VRP (Prins, 2004), before its generalization to the VFMP and HVRP in Sections 3.2 and 3.3. *Split* for the VRP builds an acyclic auxiliary graph $H = (X, A)$ with n+1 nodes. X contains one dummy node 0, plus nodes 1 to n for customers C_1 to C_n. One arc $(i-1, j)$ is included in A for each subsequence $(C_i, C_{i+1}, \ldots, C_j)$ of C if a trip serving these clients is feasible, i.e., if the total demand W_{ij} of the subsequence (trip load) does not exceed vehicle capacity

$$W_{ij} = \sum_{k=i}^{j} q(C_k) \leqslant Q \tag{1}$$

The cost Z_{ij} of the subsequence for the auxiliary graph is nothing but the length L_{ij} of the associated trip

$$Z_{ij} = L_{ij} = d(0, C_i) + \sum_{k=i}^{j-1} d(C_k, C_{k+1}) + d(C_j, 0) \tag{2}$$

An optimal splitting of C is given by a minimum-cost path from node 0 to node n in H. For m feasible subsequences ($m \leqslant n(n-1)/2 = O(n^2)$), the graph can be computed in $O(m)$ using two nested loops for i and j. For one given i, the load and cost of each subsequence can be computed in $O(1)$ when j is incremented, using the values already computed for $j-1$. The shortest path can be computed in $O(m)$ too, using the version of Bellman's algorithm for directed acyclic graphs (Cormen et al., 1990).

A small example for the VRP is depicted in Fig. 1. The giant tour for chromosome C is given with demands in brackets. Dashed lines indicate possible returns to the depot. All feasible trips extractible from C are modeled by one arc in the auxiliary graph. The arcs of the shortest path (boldface) indicate the optimal partition into trips, subject to the sequence.

3.2. Evaluation for the VFMP

Split can be generalized for the VFMP. Like for the VRP, the load W_{ij} and the length L_{ij} of each trip $(C_i, C_{i+1}, \ldots, C_j)$ can be determined in $O(1)$. The trip is feasible if it can be performed at least by the largest vehicle type, i.e., if $W_{ij} \leqslant Q_t$. Let Z_{ijk} be the trip cost for vehicle type k. The cost Z_{ij} for the auxiliary graph corresponds to the cheapest compatible vehicle type

$$Z_{ij} = \min(Z_{ijk} = f_k + v_k L_{ij} : 1 \leqslant k \leqslant t \text{ and } W_{ij} \leqslant Q_k) \tag{3}$$

The best vehicle type can be found in $O(1)$ if demands are integer (which is the case for all instances from literature) and if the fleet is correlated: a vector B indexed from 1 to the total demand of customers q_{tot} can be pre-computed, in such a way that $B(W)$ gives the best vehicle type for a trip with load W. For uncorrelated fleets, an $O(t)$ loop on vehicle types must be added and the complexity of *Split* becomes $O(mt)$.

Table 2 gives an example with $t = 3$ vehicle types. The arc weight used in the auxiliary graph for a trip with load 25 and length 300 will be 6300, corresponding to the vehicle type 2.

3.3. Evaluation for the HVRP

In the HVRP, the number of vehicles of each type is limited and the splitting problem can be infeasible. Note that the data must satisfy the necessary conditions (4), which state that the demand of customers who need a vehicle of type k or larger must not

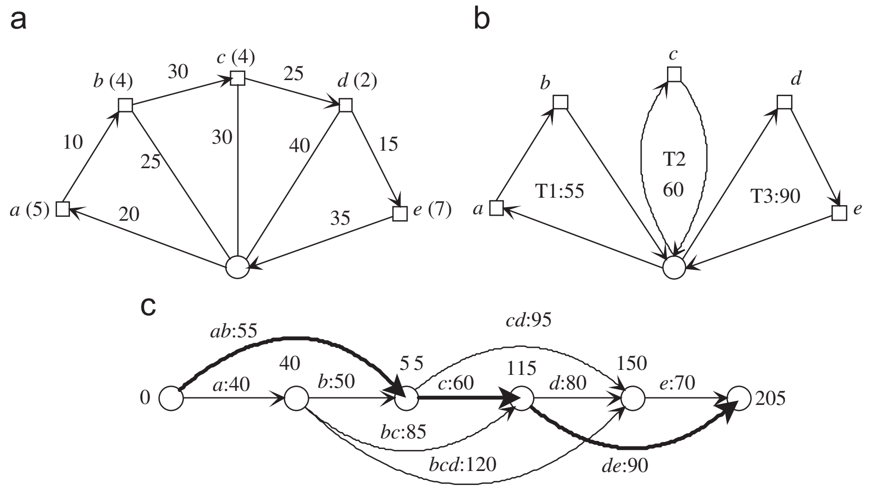


Fig. 1. *Split* applied to a small VRP example: (a) a chromosome $C = (a, b, c, d, e)$, (b) auxiliary graph for $Q = 10$ and (c) optimal splitting: 3 trips, cost 205.

Table 2
Example of trip cost computation for the VFMP.

Type k	Capacity Q_k	Fixed cost f_k	Variable cost v_k	Trip cost for load 25 and length 300
1	20	100	10	Capacity violated
2	40	300	20	$Z_{ij} = 300+20 \times 300 = 6300$
3	50	250	30	$Z_{ij} = 250+30 \times 300 = 9250$

$t = 2, a_1 = 1, Q_1 = 10, a_2 = 1, Q_2 = 14$

C	:	a	b	c	d	e	→ feasible
q	:	5	4	4	2	7	
C'	:	a	d	c	e	b	→ infeasible
q	:	5	2	4	7	4	

Fig. 2. Feasible and infeasible chromosomes.

exceed the total capacity of these vehicles. We assume that the vehicle types are sorted in increasing capacity order.

$$\sum_{i=1}^{n}(q_i|q_i\geqslant Q_k)\leqslant\sum_{p=k}^{t}a_pQ_p, \quad \text{for each vehicle type } k, \quad 1\leqslant k\leqslant t \qquad (4)$$

Fig. 2 shows a more subtle infeasible case. Chromosome C is the same as in Fig. 1 and C' is another permutation with the same customers. The demands are recalled under each customer. The fleet has only two vehicles, with respective capacities 10 and 14: contrary to C, there is no feasible splitting for C', although (4) holds. Such chromosomes can be rejected because they are rare in practice, except if the total demand is very close to fleet capacity.

The splitting problem for the HVRP is quite involved. The auxiliary graph H is the same as for the VFMP, with one arc for each trip (C_i, C_{i+1},..., C_j), such that $W_{ij}\leqslant Q_t$. The only difference is that we do not know in advance, due to limited availabilities, if one vehicle of the cheapest type will be available. Therefore, the trip length L_{ij} is used as provisional weight for the arc in H: the exact cost will depend on the type of vehicle allocated to this arc.

An optimal splitting corresponds to a least-cost path from node 0 to n in H, with no more than a_k arcs for each vehicle type k. This problem is a *shortest path problem with resource* constraints (SPPRC): each vehicle type can be viewed as a resource available in a_k units and each arc requires one unit of a compatible vehicle type. In general, such problems are NP-hard but can be solved fairly quickly in practice, using dynamic programming (DP) methods.

Let $P(j,x_1,x_2,\ldots,x_t)$ be the optimal splitting cost for subsequence $(C_1,C_2,\ldots,C_j)$, $1\leqslant j\leqslant n$, using a fleet mix with $0\leqslant x_k\leqslant a_k$ vehicles for each type k, k = 1, 2,..., t. This splitting problem can be defined by the following recursive equations:

$$P(0,x_1,x_2,\ldots,x_t)=0, \quad 1\leqslant k\leqslant t, \quad 0\leqslant x_k\leqslant a_k \qquad (5)$$

$$\forall j>0, \quad P(j,x_1,\ldots,x_t)=\min(P(i,x_1,\ldots,x_k-1,\ldots,x_t) + Z_{ijk} : i\leqslant j, \quad W_{ij}\leqslant Q_k, \quad 0<x_k\leqslant a_k) \qquad (6)$$

We now describe one possible implementation for this DP method and analyze its computational complexity. Let ψ be the maximum number of distinct vectors x, taking null consumptions into account

$$\psi=\prod_{k=1}^{t}(a_k+1) \qquad (7)$$

The computations can be organized using a matrix of labels Y, $(n+1)\times\psi$. Each row j of Y is indexed from 0 to $\psi-1$ and contains one label for each possible vector x. In practice, one label Y_{jh} for j represents a shortest path from node 0 to node j in the auxiliary graph. Contrary to the VRP and VFMP, several labels may be used for the same node, with different resource consumptions. A label can be defined by one record with one vector $Y_{jh}.x$ (the partial fleet used for the path), and a cost $Y_{jh}.Z = P(j, x_1, x_2,\ldots, x_t)$. A label can be accessed in $O(1)$ by defining a key $h(x)$ which converts each possible vector x into a column subscript between 0 and $\psi-1$ included. This key can be computed in $O(t)$ using the following recursive definition:

$$h(x,0)=0 \qquad (8)$$

$$h(x,k)=h(x,k-1)(a_k+1)+x_k, \quad \text{for } 0\leqslant k\leqslant t \qquad (9)$$

In practice, the non-recursive Algorithm 1 can be used. For instance, if $t=3$ and $a=(2,3,1)$, we have $\psi=3\times4\times2=24$, $h(0,0,0)=0$, $h(2,1,1)=19$ and $h(2,3,1)=23$.

```
01. $h := 0$
02. for $k := 1$ to $t$ do $h := h\times(a_k+1)+x_k$ endfor
```

Algorithm 1. Computation of a key for one vector x.

Algorithm 2 gives an implementation of the DP method, based on matrix Y. At the end, the solution is given by the cheapest label in row n of Y. If all labels of row n have still an infinite cost, the problem is infeasible.

All labels are initialized in line 1: all vectors x are set to 0, the costs of labels in row 0 are set to 0 and the costs of other labels to infinity. This can be done in $O(n\psi t)$. Lines 2 to 5 inspect each node i, the m arcs (i,j) of H, each label Y_{ip} used by i, and each possible vehicle type k. Hence, the cumulated number of iterations of loops of lines 2–5 is $O(m\psi t)$. This complexity dominates the previous one, which can be neglected.

Lines 6–12 compute a label U for the path obtained by adding arc (i, j) with vehicle k to the incumbent label/path defined by label Y_{ip}. U is initialized with a copy of Y_{ip}, $U.x_k$ is incremented and the cost of arc (i, j) for vehicle type k is added to $U.Z$. Finally, the key h for vector $U.x$ is computed and label Y_{jh} is replaced by U in case of improvement. All statements in lines 6–12 are in $O(1)$, except the copy of Y_{ip} into U and the calculation of h, both in $O(t)$. Hence, the algorithm is in $O(m\psi t^2)$.

```
01. initialize matrix $Y$
02. for $i := 0$ to $n$ do
03.   for each arc $(i, j)$ of $H$ do
04.     for $p := 0$ to $\Psi-1$ with $Y_{ip}.Z < \infty$ do
05.       for $k := 1$ to $t$ with $(Y_{ip}.x_k<a_k)$ and $(W_{ij}\leqslant Q_k)$ do
06.         $U := Y_{ip}$
07.         $U.x_k := U.x_k+1$
08.         $U.Z := U.Z+f_k+v_k\cdot L_{ij}$
09.         compute the key $h$ for the consumption vector $U.x$
10.         if $U.Z<Y_{jh}.Z$ then
11.           $Y_{jh} := U$
12.         endif
13.       endfor
14.     endfor
15.   endfor
16. endfor
```

Algorithm 2. Splitting algorithm for the HVRP.

In fact, the key in line 9 can be computed in $O(1)$ if we keep with each label the associated key. To get the key for the new vector $U.x$ from the one stored in Y_{ip} for x, we just add the contribution π_k of a vehicle of type k

$$\pi_k=\prod_{i=1}^{k-1}(a_i+1) \qquad (10)$$

These contributions can be pre-computed at the beginning in a vector π. Using this trick, it is important to note that we *do not* calculate the new vector x explicitly, which would cost $O(t)$, but only its cost and its key in $O(1)$. The DP method has then a pseudo-polynomial complexity in $O(m\psi t)$. However, the fleet has at most n vehicles (one per customer), which implies that the number of vehicles for each type $k(a_k)$ is at most n and $\psi = O(n^t)$ from Eq. (7). We have proved the following:

Theorem. *If the total number of vehicles does not exceed the number of customers, the splitting problem for the HVRP can be solved in $O(mtn^t)$.*

This complexity depends on the number of vehicle types t, but not on the total number of vehicles. Moreover, if the number of vehicle types t is stable during a long period, the algorithm is fully polynomial. In practice, *Split* for the HVRP is fast enough because m is much smaller than n^2 (most long subsequences are infeasible) and the number of vehicle types is small ($t = 3$–6 in the Golden et al. instances).

Our implementation is strongly accelerated on average using dominance and capacity tests. Label $L = (Z, x)$ *dominates* label $L' = (Z', x')$ if $Z \leqslant Z'$ and $\forall k{:}x_k \leqslant x'_k$, with at least one strict inequality. This means that L is at least as good as L' in terms of cost, but consumes fewer resources. This test can be used to keep in Y non-dominated labels only. This idea was used by Desrosiers et al. (1988) in a DP method for the shortest path problem with time windows.

Concerning the capacity test, let R_p be the remaining demand of sequence C from index p onward, i.e., for $(C_p, C_{p+1},\ldots, C_n)$, and $F(x)$ the residual capacity of the fleet for a given consumption vector x. The values of R_p, $p = 1, 2,\ldots, n$ can be pre-computed at the beginning. The value for F can be kept in each label and deduced in $O(1)$, for each new label. Obviously, vehicle type k can be ignored in line 4 if $R_{i+1} > Y_{ip}.F–Q_k$. If vehicle types are sorted in increasing order of capacity, we can even stop the *for* loop as soon as this test fails.

3.4. Population structure

The population is stored in a table P of nc chromosomes (giant tours). The initial population is filled with random permutations of customers and the first three giant tours are improved using the local search of Section 3.6. Hence, P contains three good solutions to accelerate the search and nc–3 low-quality solutions to bring more diversity. When loading data, the algorithms check whether each vehicle type provides enough capacity to satisfy the total demand. If *yes*, the instance is a VFMP, otherwise it is an HVRP. This automatic recognition allows calling the ad hoc version of *Split* to evaluate chromosomes. At the beginning and at any iteration, P is kept sorted in an ascending cost order: the current best solution is always P_1.

3.5. Basic iteration

In any incremental genetic algorithm, the basic iteration consists in selecting two parents, applying a crossover operator and replacing existing solutions by the offspring. In MAs, new solutions are improved using a local search. Two local search procedures for the VFMP and HVRP are described in the next subsection.

At each iteration, two parents A and B are selected using the binary tournament method: two chromosomes are randomly drawn using a uniform distribution, the best one is kept for A, and the same process is repeated to get B.

The giant tour encoding allows recycling crossovers designed for the TSP. We selected the *order crossover* or OX (Oliver et al., 1987), see Fig. 3 for one example. OX randomly draws two positions i and j in A. The substring between i and j (included) is copied into child C_1, at the same positions. B is finally browsed cyclically, from position j+1 onwards, and its customers not yet present in the child are used to complete C_1 cyclically, starting also from position j+1. The roles of the two parents are exchanged to get the other child C_2. The original OX generates two children-solutions. In our version, one child C is randomly selected.

			i ↓		j ↓		
Parent A	7	3	4	1	5	6	2
Parent B	1	2	3	4	5	6	7
Child $C1$	2	3	4	1	5	6	7
Child $C2$	7	1	3	4	5	6	2

Fig. 3. Example of OX crossover.

Child C is evaluated by *Split*, and replaces one chromosome P_r randomly selected in the worst half of the population P, i.e., among the nc/2 last chromosomes. Finally, C is shifted to keep the population sorted in an increasing cost order.

insertion sort

3.6. Local search

In Moscato's MA model (1999), each new solution undergoes a local search. This is not systematic in our algorithms, in which the local search is applied with a given probability p_{LS} to the child C resulting from a crossover. The procedure works on complete solutions computed by *Split* instead of giant tours. The moves include the relocation of one customer, the exchange of two customers and 2-OPT moves. They are applied to one or two routes.

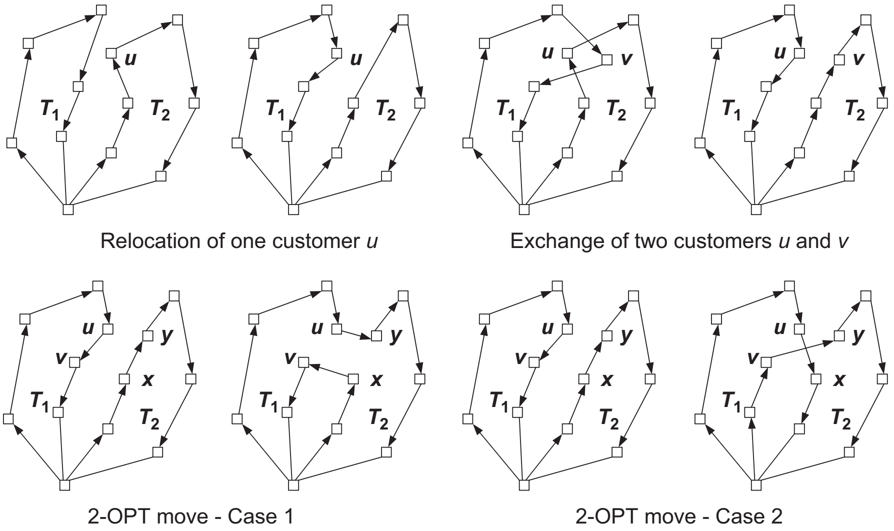
Fig. 4 gives one example of each move for two trips. For one trip, the well-known 2-OPT move replaces two non-adjacent arcs (u, v) and (x, y), traversed in this order, by arcs (u, x) and (v, y), with inversion of the trip segment from v to x included. For two trips, there are two possibilities to reconnect trip segments: the removed arcs (u, v) and (x, y) are replaced either by (u, y) and (x, v) or by (u, x) and (v, y). The second case requires the inversion of some segments.

Each local search iteration scans the proposed moves in $O(n^2)$ and performs the first improving move. The process is repeated until no improvement is found. The trips are finally concatenated into a chromosome (giant tour), which is re-evaluated by *Split*. *Split* behaves like an additional improvement operator, able to shift all trip limits and to reshuffle vehicles.

Two versions of the local search have been tested. The first one LS_1 is not allowed to change the vehicle-to-trip assignment. It computes the new length of each trip affected by a move and deduces the new cost from the fixed and variable costs of the vehicle currently assigned to the trip. The fixed cost is removed if the trip becomes empty. Note that the vehicles, fixed during the local search, can be changed by *Split* which is called to re-evaluate the resulting solution.

The second local search LS_2 is more powerful and may change vehicles. Consider a VFMP solution and a move affecting one trip T_1 with cost C_1 and one trip T_2 with cost C_2. To evaluate this move, the new trip loads W'_1 and W'_2 and the new lengths L'_1 and L'_2 can be computed in $O(1)$. The cheapest compatible vehicle types k and p for W'_1 and W'_2 can be computed in $O(1)$ too if the fleet is correlated, in $O(t)$ otherwise (see Section 3.2). An improving move is detected if the cost variation $(f_k+v_k \cdot L'_1+f_p+v_p \cdot L_2')–(C_1+C_2)$ is negative.

For the HVRP, let F be the set of free vehicles and u, v the vehicles used by T_1 and T_2. LS_2 tries four changes of vehicles: (a) the two trips exchange their vehicles, (b) T_1 gives its vehicle to T_2 and takes one in F, (c) T_2 gives its vehicle to T_1 and takes one in F, (d) both T_1 and T_2 exchange their current vehicle with a free one.



Relocation of one customer *u*

Exchange of two customers *u* and *v*

2-OPT move - Case 1

2-OPT move - Case 2

Fig. 4. Examples on two trips for the three moves.

In fact, many HVRP instances from literature use almost all vehicles and *F* is often empty. In such cases, changes of vehicles during the local search are rare and LS_1 is better suited: it is almost as good as LS_2, but faster.

4. Two MAs

Two MAs based on the components of Section 3 are proposed. The traditional mutation operator is replaced by a solution dispersal mechanism. In the sequel, we assume that each element P_k in population *P* is a record with three fields: one chromosome (giant tour) $P_k.T$, the solution $P_k.S$ deduced by *Split*, and its cost $P_k.F$. To simplify algorithms, it is assumed that the call *Split* (P_k, *feasible*) evaluates the giant tour $P_k.T$ and returns a feasibility flag, always *true* for the VFMP. If *feasible* = *true*, the resulting solution is stored in $P_k.S$ and its cost in $P_k.F$. The call $LS_1(P_k)$ applies the local search to the solution $P_k.S$ and includes the re-evaluation mentioned in Section 3.6: on output, the trips of the improved solution are concatenated to form a new chromosome $P_k.T$, which is re-evaluated by *Split*.

4.1. First MA

The first MA, called FMA, uses a dispersal rule in objective space: any two solutions in *P* must have a difference in cost not smaller than a given value Ω, i.e., $|P_i.F - P_j.F| \geqslant \Omega$, for any two records *i* and *j*. This rule is enforced in the initial population and for each new solution *C*: the solution P_r selected for substitution is replaced by *C* only if $P \backslash \{P_r\} \cup C$ satisfies the rule. However, to avoid loosing a new best solution, the rule is overridden if $C.F < P_1.F$. Contrary to standard MAs, a child may be discarded because of this rule, giving an unproductive iteration. However, the rejection rate is low (below 10%) if the population is not too large, 50 chromosomes for instance. FMA uses the basic local search LS_1, the one which does not change the vehicle assigned to each trip.

Algorithm 4 gives an outline of FMA. The initial population is built in lines 1–7. The *repeat* loop tries random candidates for the incumbent chromosome P_k, until one satisfies the dispersal rule line 6. Each candidate is evaluated by *Split* which stores the associated solution in $P_k.S$ and its cost in $P_k.F$. The local search LS_1 is applied to the three first solutions.

The main loop covers lines 8–24. It performs *np* phases. Each phase starts by sorting the population in increasing order of costs, performs *ni* basic iterations (lines 10–22) and ends by a partial renewal of the population (line 23): the procedure *Partial_Renewal* keeps the *nk* best solutions and replaces the *nc*–*nk* others by new random solutions.

Each basic iteration selects two parents *A* and *B* using the binary tournament method and applies the OX crossover to get a giant tour *C* evaluated by *Split*. If feasible, the new solution undergoes local search with a fixed probability p_{LS} and replaces a chromosome P_r randomly selected in the worst half of *P* if the dispersal rule is satisfied or (exception) if it outperforms the current best solution. At the end, the best solution is in P_1.

```
01. for $k := 1$ to $nc$ do
02.   repeat
03.     generate a random giant tour in $P_k.T$
04.     Split ($P_k$, feasible)
05.     if (feasible) and ($k \leqslant 3$) then $LS_1(P_k)$ endif
06.   until (feasible) and ($\forall\, i<k: |P_i.F - P_k.F| \geqslant \Omega$)
07. endfor
08. for $p := 1$ to $np$ do
09.   sort $P$ in increasing cost order
10.   for $i := 1$ to $ni$ do
11.     select two parents $A$ and $B$ by binary tournament
12.     apply crossover OX to $A$ and $B$ to give one child $C$
13.     Split (C, feasible)
14.     if feasible then
15.       if random $< p_{LS}$ then $LS_1(C)$ endif
16.       randomly select $P_r$ for replacement, in the worst half of $P$
17.       if ($C.F < P_1.C$) or ($\forall\, i \neq r: |P_i.F - C.F| \geqslant \Omega$) then
18.         $P_r := C$
19.         shift $P_r$ to keep $P$ sorted
20.       endif
21.     endif
22.   endfor
23.   Partial_Renewal ($P$, $nk$)
24. endfor
```

Consider using near neighbor

Algorithm 3. General structure of FMA.

4.2. Second MA

The second algorithm called SMA is a *MA with population management* or MA|PM, a new structure of MA recently introduced by Sörensen and Sevaux (2006). MA|PM controls solution diversity using a distance measure in solution space. Let $d(A, B)$ be the distance between two solutions A and B and $D(P, C) = \min\{d(A, C): A \in P\}$ the distance of a new solution C to population P. C is accepted if and only if $D(P, C) \geqslant \Delta$, where Δ is a non-negative threshold.

When the distance test fails, Sörensen and Sevaux (2006) recommend to apply a mutation operator until the child is accepted. However, mutation has sometimes a weak effect on the distance: the repeated mutations are time-consuming and can lead to poor-quality solutions. Like in FMA, we prefer to discard the offspring which does not satisfy the dispersal criterion.

We have tested several distances but the one giving the best results on average is the *distance for R-permutations* (Campos et al., 2005) or *broken pairs distance*. For two chromosomes A and B, this distance $d(A, B)$ is the number of pairs of adjacent customers in A which are broken (i.e., no longer adjacent) in B. For instance, if $A = (1, 2, 3, 4, 5, 6)$ and $B = (4, 5, 6, 2, 3, 1)$, (1, 2) and (3, 4) are broken in B and $d(A, B) = 2$. This distance has integer values from 0 to n–1 included and can be computed in $O(n)$.

The only difference between SMA and Algorithm 3 are the following:

- The local search LS_1 must be replaced by LS_2 in lines 5 and 15.
- To get distinct initial solutions, P_k is accepted in line 6 if $D(\{P_1,\ldots, P_{k-1}\}, P_k) \neq 0$. Child C is accepted in line 17 if $C.F < P_1.C$ or $D(P\backslash\{P_r\}, C) \geqslant \Delta$.
- The simplest policy for Δ is to use a constant value initialized at the beginning. We tried two strategies, which give better results on average. In the *increasing policy*, Δ grows linearly from a given value $\Delta_{\min}$ to a given value $\Delta_{\max}$ during the *ni* iterations of each phase. In the *decreasing policy*, Δ decreases linearly from $\Delta_{\max}$ to $\Delta_{\min}$ in each phase. $\Delta_{\max}$ can be as large as the maximum value of the distance for R-permutations, $n-1$. However, children reaching this distance to the population are very rare. To avoid too many unproductive iterations, $\Delta_{\max}$ is limited in practice to a fraction of the maximum distance, e.g., $0.5 \times (n-1)$.

5. Computational evaluation

5.1. Implementation, instances and methods compared

The two MAs FMA and SMA were implemented in Delphi and evaluated on a Dell Latitude laptop PC with a 1.8 GHz Pentium 4M and Windows 2000. Euclidean distances and trip costs were computed using double precision real numbers. Like in all published papers, solution values given in our tables are rounded to the closest second decimal.

The 20 instances from Golden et al. (1984), briefly presented in Section 2, were used to compare our results with those of the best published algorithms. They can be found for instance at http://or.ingce.unibo.it/research/hvrp. All fleets are correlated and include two to six vehicle types. To save space, we do not recall the fleet composition for each problem, see for instance the tables in Li et al. (2007) or Choi and Tcha (2007). Recall that instances 1, 2 and 7 to 12 are given with a distance matrix, while all other instances include coordinates from which Euclidean distances can be deduced.

The original instances had no variable costs. These costs were added by Taillard (1999) in instances 13–20 and later by Choi and Tcha (2007) in instances 3–6. VFMP-F instances are obtained by setting all variables costs v_k to 1. VFMP-V instances are deduced by setting all fixed costs f_k to zero. VFMP-FV instances use the two kinds of cost. The true HVRP instances are based on instances 13–20, using limited numbers of vehicles defined by Taillard *and ignoring fixed costs*.

The kind of algorithm and instances used in published papers have been already given in Table 1 of Section 2. Our algorithms were compared with the true metaheuristics of this table, i.e., without the heuristics of Golden et al. (1984) and of Renaud and Boctor (2002). Moreover, to evaluate the deviations to the bounds of Choi and Tcha (2007), which are valid for Euclidean instances only, we excluded the non-Euclidean instances 1–2 and 7–12.

5.2. Results for the VFMP with fixed costs only

FMA and SMA were compared on the Euclidean VFMP-F instances 3–6 and 13–20 with the HCG of Taillard (1999), two tabu search methods (Gendreau et al., 1999; Wassan and Osman,

Table 3
Parameters used for FMA and SMA for VFMP instances F, V and FV.

Method	*nc*	*np*	*ni*	*nk*	*LS*	P_{LS}	Ω	Policy	$\Delta_{\min}$	$\Delta_{\max}$	Flat
FMA-1	30	10	3000	1	1	0.1	0.5	–	–	–	–
SMA-D1	30	5	4000	1	2	0.2	–	Down	1	$0.2(n-1)$	1000
SMA-U1	30	5	3000	1	2	0.2	–	Up	1	$0.5(n-1)$	0
SMA-U2	30	5	4000	1	2	0.2	–	Up	1	$0.5(n-1)$	0

Table 4
Summary of results for Euclidean VFMP-F instances 3–6 and 13–20.

Method	Best run					Average		
	Dev. to BKS %	Dev. to LB %	Scaled time (s)	BKS found	BKS improved	Dev. to BKS %	Dev. to LB %	Scaled time (s)
Choi and Tcha	0.002	1.654	159.1	11	0	0.115	1.768	217.9
SMA-D1	0.040	1.692	42.5	7	1	0.169	1.822	40.7
SMA-U2	0.051	1.703	36.0	6	1	0.189	1.843	41.5
SMA-U1	0.104	1.756	26.1	7	0	0.225	1.879	31.0
Taillard	0.139	1.792	–	6	0	0.511	2.370	45.7
FMA-1	0.145	1.799	13.5	6	1	0.324	1.980	16.6
Wassan and Osman	0.233	1.887	12.6	5	0	–	–	–
Gendreau et al.	0.240	1.895	13.2	5	0	0.428	2.085	–
Lima	0.306	1.962	–	5	0	0.566	2.226	1022.2

2002), the MA of Lima et al. (2004) and the B&B algorithm of Choi and Tcha (2007).

Table 3 indicates the parameters used by our algorithms and mentioned in Section 4. The first four are the number of chromosomes (*nc*), the number of phases (*np*), the number of iterations per phase (*ni*) and the number of solutions kept in the partial renewal at the end of each phase (*nk*). So, the total number of crossovers executed is $np \times ni$. The two next columns concern the local search (see subSection 3.6): the type 1 or 2 (*LS*) and the rate (P_{LS}). Column Ω indicates the cost spacing used only by FMA.

The other columns concern the algorithm with population management, SMA: the policy for Δ (*up* = increasing, *down* = decreasing) and the minimum and maximum values of Δ. The last column indicate a possible flat area at the end of each phase, which often gives better results for the down policy: Δ decreases linearly from $\Delta_{\max}$ to $\Delta_{\min}$ in the *ni–flat* first iterations of each phase, followed by *flat* iterations with $\Delta = \Delta_{\min}$. The different combinations of parameters define one version for FMA (FMA-1) and three for SMA: one with the decreasing policy (SMA-D1) and two with the increasing one (SMA-U1 and SMA-U2).

Table 4 provides some statistical results, while detailed results are given in Appendix B. Columns 2–6 indicate the deviation to the best-known solutions (BKS) in percent, the deviation to the LB of Choi and Tcha (2007) in percent, the running time in seconds (scaled for our 1.8 GHz PC), the number of BKS retrieved and the number of BKS improved. The last columns give deviations and the scaled time, averaged over different runs. All deviations and times concern instances 3–6 and 13–20, except the figures given for Taillard in columns 7–9, which ignore instances 3–6. Dashes indicate missing or irrelevant information.

Columns 2–6 concern the best result of 5 runs, except for Gendreau et al. and Lima et al., who used 10 runs. The best-known solutions (BKS) take into account papers until Choi and Tcha (2007). However, we had to correct some errors. Wassan and Osman (2002) provide detailed solutions, which are visibly computed with two decimals. We recomputed their costs in double precision and found 2730.10 vs. 2730.08 for instance 13 and 8665.75 vs. 8659.74 for instance 19. The BKS given in Choi and Tcha for instance 19 (8659.74) must then be replaced by 8661.81, found by Taillard in 1999. Table 2 in Gendreau et al. (1999) contains typos, but correct values are given with detailed solutions in Appendix A of their paper. The correct values are 6516.47 instead of 6516.46 for instance 6, 2408.62 instead of 2408.41 for instance 13, and 9119.28 instead of 9119.03 for instance 14.

The original running times were scaled using the performances of computers listed in Dongarra (2006). Taillard and Gendreau et al. used a 50 MHz Sun Sparc 10, Wassan and Osman a 50 MHz Sun Sparc 1000, Lima et al. a 2 GHz Pentium 4 and Choi and Tcha a 2.6 GHz Pentium 4. According to Dongarra, these computers have respective powers of 27, 25, 1737 and 2266 Mflop/s, vs. 1564 Mflop/s for our 1.8 GHz Pentium 4M PC. The scaled times are relatively precise for different PCs, for which they are roughly proportional to the inverse of clock frequency. They provide only an approximation for the older Sun computers, mainly because modern compilers generate a more efficient code.

Table 4 is sorted in the increasing order of deviations to the best-known solutions. All our algorithms outperform the early MA of Lima et al. Our simplest algorithm, FMA-1, does better than the two TS, while being as fast. The versions with population management are much better but need larger running times, which can be explained by the computation of distances and the fraction of children-solutions rejected by the distance test. Our best metaheuristic is an MA|PM with decreasing policy, SMA-D1. All our MAs except SMA-U1 improve one best-known solution. New best solutions are given in Appendix A. The B&B method of Choi and Tcha is still the most efficient algorithm, but also the slowest one.

Table 5
Summary of results for Euclidean VFMP-V instances 13–20.

Method	Best run					Average		
	Dev. to BKS%	Dev. to LB%	Scaled time (s)	BKS found	BKS improved	Dev. to BKS%	Dev. to LB%	Scaled time (s)
Choi and Tcha	0.000	2.086	78.0	8	0	0.142	2.231	117.5
SMA-U2	0.034	2.120	26.5	6	1	0.162	2.251	31.6
SMA-U1	0.035	2.121	28.0	6	1	0.165	2.254	33.9
SMA-D1	0.039	2.125	48.1	7	0	0.107	2.195	49.2
Gendreau et al.	0.088	2.175	19.9	7	0	0.602	2.699	–
FMA-1	0.261	2.352	38.6	4	0	0.720	2.818	46.5
Wassan and Osman	0.462	2.558	22.7	2	0	–	–	–
Taillard	0.773	2.873	–	1	0	–	–	34.7

Table 6
Summary of results for Euclidean VFMP-FV instances 3–6 and 13–20.

Method	Best run					Average		
	Dev. to BKS %	Dev. to LB %	Scaled time (s)	BKS found	BKS improved	Dev. to BKS %	Dev. to LB %	Scaled time (s)
SMA-U1	−0.063	1.497	23.1	7	1	0.018	1.580	25.6
SMA-D1	−0.013	1.549	44.5	6	2	0.038	1.601	42.6
Choi and Tcha	0.000	1.562	134.3	12	0	0.030	1.592	159.9
SMA-U2	0.005	1.567	25.3	6	2	0.032	1.595	28.1
FMA-1	0.012	1.574	17.2	5	1	0.064	1.627	16.7

5.3. Results for the VFMP with variable costs only

Choi and Tcha (2007) added variable costs to VFMP-F instances 3–6, but the resulting VFMP-V files have not yet been used by other authors. The comparison is then done with instances 13–20 and involves the same algorithms as in the previous subsection, except the MA of Lima et al. In particular, we reuse our versions FMA-1, SMA-U1, SMA-U2 and SMA-D1, based on parameters of Table 3. Each metaheuristic performs the same number of runs as in §5.2, except the TS of Wassan and Osman which is executed twice instead of five times. Two errors in Wassan and Osman were corrected: the exact cost is 1108.97 and not 1100.56 for file 19, 1536.24 and not 1530.16 for file 20. The best solutions in Choi and Tcha become 1105.44 for file 19 (found by Gendreau et al.) and 1530.43 for file 20 (found by Choi and Tcha).

Table 5 provides some key-figures. The MAs are strongly improved by population management. SMA-U1 and U2 are even faster than FMA-1 in spite of the overhead induced by distance computations, indicating a faster convergence. They also find a new best solution. Contrary to the VFMP-F, FMA-1 is superseded here by the TS from Gendreau et al. The B&B algorithm of Choi and Tcha outperforms again our best MA (here SMA-U2) in terms of average deviation, but it is still slower.

5.4. Results for the VFMP with fixed and variable costs

Our MAs FMA-1, SMA-U1, SMA-U2 and SMA-D1 defined by the parameter settings of Table 3 are compared with the only published method tested on VFMP-FV instances 3–6 and 13–20, the B&B algorithm from Choi and Tcha (2007). All methods are executed five times. Table 6 shows that all the MAs improve one or two best-known solutions. This time, two versions with population management outperform the B&B method. The best algorithm, here SMA-U1, is even six times faster.

5.5. Results for the HVRP

Taillard (1999) was the first to define HVRP instances by adding limited numbers of vehicles and variables costs (but no fixed costs) to instances 13–20 from Golden et al. The only other metaheuristics tested on these instances are threshold accepting algorithms (Tarantilis et al., 2003, 2004) and a RTR travel method (Li et al., 2007). Table 7 gives the parameters used by our algorithms, defining three versions. Compared to Table 3, we can see that the HVRP require higher local search rates. As explained in Section 3.6, most vehicles are used in all instances and LS_2 finds too few changes of vehicles: the faster LS_1 can be used.

Table 7
Parameters used for FMA and SMA for HVRP instances.

Method	*nc*	*np*	*ni*	*nk*	*LS*	P_{LS}	*Ω*	Policy	$\Delta_{\min}$	$\Delta_{\max}$	Flat
FMA-2	30	5	3000	1	1	0.25	0.5	–	–	–	–
SMA-D2	30	5	4000	1	1	0.50	–	Down	1	$0.5(n-1)$	1000
SMA-U3	30	5	3000	1	1	0.20	–	Up	1	0.5(n−1)	0

Some results are given in Table 8, see Appendix B for details. Like Tarantilis et al. and Li et al., and contrary to VFMP instances, we performed a single run of our algorithms. However, Taillard used five runs and gave the best deviations of these runs, but an average running time. The best-known solutions are the ones listed in Li et al. (2007). Choi and Tcha (2007) designed no LB for the HVRP, but we took their bound for the VFMP-V, still valid yet weaker for the HVRP. Dongarra (2006) report a power of 262 Mflops/s for the 400 MHz Pentium II used by Tarantilis et al. and 1168 Mflops/s for the 1 GHz Athlon PC used by Li et al. These powers were used to scale the original running times for our 1.8 GHz PC.

Table 8 indicates that our MAs outperform the HCG of Taillard (1999) and the threshold accepting method of Tarantilis et al. (2004). Compared with our best MA (SMA-D2), the recent RTR travel algorithm from Li et al. achieves a smaller deviation, but this is due to one instance only (HVRP-17), as shown in Table 12. Moreover, the RTR algorithm is twice slower. Table 12 also shows that nobody has been able to find the best solution computed by Taillard in 1999 for HVRP-19 (1117.51).

6. Conclusion

This paper proposes for the first time a unified approach for solving vehicle routing with heterogeneous fleets. Contrary to the published methods which are limited to some versions, our algorithms are able to tackle the VRP, the three versions of the VFMP (with fixed costs, variable costs or both) and the HVRP. Moreover, they are not limited to correlated fleets.

Our algorithms are rather simple. Both use solutions encoded without trip delimiters, which can be evaluated with a splitting procedure. The evaluation is polynomial for the VRP and VFMP and pseudo-polynomial, but fast in practice for the HVRP. The first algorithm FMA is clearly improved if a distance measure in solution space is added, to give an MA|PM. Our best version is outperformed only by a B&B procedure on VFMP-F and VFMP-V instances, and by a RTR travel algorithm on HVRP instances. However, it becomes the best metaheuristic on VFMP-FV and it is faster than the best competitor for all problems. Moreover, six new best solutions have been discovered.

Table 8
Summary of results for Euclidean HVRP instances 13–20.

Method	Dev. to BKS %	Dev. to LB %	Scaled time (s)	BKS found	BKS improved
Li et al.	0.032	3.400	213.4	7	0
SMA-D2	0.077	3.447	94.8	6	0
SMA-U3	0.315	3.692	65.7	2	0
FMA-2	0.486	3.868	73.6	2	0
Tarantilis et al.	0.617	4.003	101.7	1	0
Taillard	0.931	4.326	34.7	1	0

Appendix A. New best solutions

Note: Except for VFMP-F17, these new best solutions were already found by a preliminary MA presented at Odysseus, 2006 (Labadi et al., 2006).

```
VFMP-F17, 8 trips, cost 1737.934.
Customers per trip, vehicle type, trip cost, list of customers:
 8  3  256.188  53  14   59  11  66  65  38  12
 2  1   40.456   4  75
14  3  295.765  51   3   44  50  25  55  18  24  49  56  23  63  16  68
11  3  247.438   2  62   22  64  42  41  43   1  73  33   6
 9  3  245.595  17  40   32   9  39  72  31  10  58
15  3  284.963   8  19   54  13  57  15  37  20  70  60  71  69  36   5  29
10  3  244.646  52  27   45  48  47  21  61  28  74  30
 6  2  122.882  26   7   35  46  34  67


VFMP-F18, 16 trips, cost 2369.646
 7  3  181.410   8  35   19  54  13  57  27
 2  2   66.324  12  40
 6  4  277.748  14  59   66  65  10  58
 5  3  177.289  62  22   61  28   2
 2  2   70.626   6  33
 2  2   58.416  34  46
 4  3  165.733  38  11   53   7
 2  2   65.700   4  52
 2  2   52.125  26  67
10  4  315.813  16  49   24  18  50  25  55  31  39  72
11  4  295.133  74  21   47  36  69  71  60  70  20  37  15
 1  1   16.000  75
 5  3  153.782  30  48    5  29  45
 1  1   26.125  17
 5  3  159.956  51   3   44  32   9
10  4  287.466  68  73    1  43  42  64  41  56  23  63


VFMP-V18, 7 trips, cost 1800.802
 4  2  131.832  72  31   55  25
15  5  303.092  75  30   48  47  36  69  71  60  70  20  37   5  29  45   4
22  6  595.253  51   3   44  32  50  18  24  49  23  56  41  43  42  64  22  62  28  61  21  74   2  68
 7  3  153.878   8  19   54  13  57  15  27
17  6  440.220  26   7   35  53  14  59  11  66  65  38  10  58  39   9  40  12  17
 4  3   58.629  34  52   46  67
 6  3  117.898  16  63    1  73  33   6


VFMP-FV17, 12 trips, cost 2004.481
 7  2  163.065  46   8   35  19  54  27  52
 7  2  168.256  48  47   36  37   5  29  45
 9  2  210.627  21  69   71  60  70  20  15  57  13
 6  2  170.102  30  74   61  28  62   2
 8  3  302.841  38  65   66  11  59  14  53   7
 7  2  172.397  51  16   23  56  63  33   6
 6  2  141.579  17   3   44  40  12  26
 7  2  211.539  49  24   18  55  25  50  32
 6  2  184.295  58  10   31   9  39  72
 8  2  193.555  73   1   43  41  42  64  22  68
 2  1   40.456   4  75
 2  1   45.770  34  67


VFMP-FV19, 15 trips, cost 8662.856
 5  1  563.886  26  54    4  72  21
 5  1  539.994  94  59   92  95  13
 9  1  626.543  24  29   78  34  35  65  71   9  51
 6  1  560.522  18  82   48   7  52  27
 8  1  602.498   2  57   15  43  38  14  42  87
 7  1  623.582   8  46   47  36  49  64  19
 9  1  572.532  89  60   83  45  17  84   5  99  96
 5  1  552.681  93  85  100  37  97
 7  1  578.754  88  62   11  63  90  10  31
 6  1  601.888  55  25   39  67  23  56
 5  1  546.699  12  80   68  76  28
 6  1  560.065  50  33   81  79   3  77
 7  1  591.171  69  70   32  66  20  30   1
 7  1  576.493   6  61   16  86  44  91  98
 8  1  565.548  58  41   22  75  74  73  40  53
```

```
VFMP-FV20, 25 trips, cost 4153.113
 3   1   147.572   98   85    99
 5   1   208.261   78   34    35   65   71
 4   1   158.775   93   61   100   37
 5   1   174.578   50    3    29   24   54
 4   1   177.937   55   25    39    4
 4   1   162.498   83   45    84    5
 6   1   173.042   87   42    43   15   57    2
 5   1   165.932   41   22    75   74   21
 4   1   178.198   16   86    17   60
 3   1   159.466   33   81    79
 3   1   135.409   97   95    94
 4   1   139.308   59   96     6   89
 3   1   147.793   80   68    77
 3   1   126.834   13   58    53
 3   1   192.350   56   23    67
 4   1   170.821   70   32    30    1
 5   1   186.290   91   44    38   14   92
 4   1   148.609   40   73    72   26
 3   1   157.661   52   48    18
 3   1   137.830   28   76    12
 5   1   178.658   10   90    63   11   62
 6   1   218.027    7   64    49   36   46    8
 4   1   188.135   20   66     9   51
 3   1   174.803   19   47    82
 4   1   144.326   88   31    69   27
```

Appendix B. Results instance per instance

See Appendix Tables 9–12.

Table 9
Best results found on VFMP-F instances.

Instance	*n*	BKS	Taillard		Gendreau et al.		Wassan & Osman		Lima et al.		Choi & Tcha		SMA-D1		Our best
			Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost
3	20	961.03	**961.03**	–	**961.03**	164.0	**961.03**	69.0	**961.03**	89.4	**961.03**	0.42	**961.03**	0.04	**961.03**
4	20	6437.33	**6437.33**	–	**6437.33**	253.0	**6437.33**	50.0	**6437.33**	84.5	**6437.33**	0.23	**6437.33**	0.03	**6437.33**
5	20	1007.05	1008.59	–	**1007.05**	164.0	**1007.05**	63.0	**1007.05**	84.8	**1007.05**	0.67	**1007.05**	0.09	**1007.05**
6	20	6516.47	**6516.47**	–	**6516.47**	309.0	**6516.47**	1.0	**6516.47**	84.9	**6516.47**	0.27	**6516.47**	0.08	**6516.47**
13	50	2406.36	2413.78	470.0	2408.62	724.0	2422.10	293.0	2408.60	559.3	**2406.36**	8.16	**2406.36**	17.12	**2406.36**
14	50	9119.03	**9119.03**	570.0	9119.28	1033.0	9119.86	485.0	**9119.03**	668.5	**9119.03**	35.64	**9119.03**	19.66	**9119.03**
15	50	2586.37	**2586.37**	334.0	**2586.37**	901.0	**2586.37**	1122.0	2586.88	553.9	**2586.37**	8.13	**2586.37**	25.10	**2586.37**
16	50	2720.43	2741.50	349.0	2741.50	815.0	2730.10	537.0	2721.76	506.7	**2720.43**	6.77	2729.08	16.37	**2720.43**
17	75	1744.83	1747.24	2072.0	1749.50	1022.0	1755.10	1402.0	1758.53	1516.8	**1744.83**	160.36	1746.09	52.22	**1737.93**
18	75	2371.49	2373.63	2744.0	2381.43	691.0	2385.52	1516.0	2396.47	1613.2	**2371.49**	46.36	**2369.65**	36.92	**2369.65**
19	100	8661.81	**8661.81**	12528.0	8675.16	1687.0	8665.75	2001.0	8691.00	2900.1	8664.29	890.39	8665.12	169.93	8662.86
20	100	4039.49	4047.55	2117.0	4086.76	1421.0	4061.64	1918.0	4093.29	2382.5	**4039.49**	160.64	4044.78	172.73	4041.77
Av. dev. BKS or av. time			0.139%	2648.0 s	0.240%	765.3 s	0.233%	788.1 s	0.306%	920.4 s	0.002%	109.8 s	0.040%	42.5 s	−0.034%
Scaled time for 1.8 GHz				45.7 s		13.2 s		12.6 s		1022.2 s		159.1 s		42.5 s	1.617%
Nb of BKS retrieved			6		5		5		5		11		7		8
Nb of BKS improved			0		0		0		0		0		1		2

BKS: best-known solutions taking into account published papers until Choi and Tcha (2007), see corrections done in Section 5.2.
Cost: best cost of 10 runs for Gendreau et al. (1999) and Lima et al. (2004), of 5 runs otherwise.
Time: duration from original papers, average duration of a run for Taillard (1999) and Lima et al. (2004), duration of best run otherwise.
Our best: best cost found by our memetic algorithms, using various settings of parameters.
Boldface: BKS retrieved or improved, improvements are also underlined.

Table 10
Best results found on VFMP-V instances.

Instance	*n*	BKS	Taillard		Gendreau et al.		Wassan & Osman		Choi & Tcha		SMA-U2		Our best
			Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost
13	50	1491.86	1494.58	473.0	**1491.86**	626.0	1499.69	140.0	**1491.86**	4.11	**1491.86**	3.45	**1491.86**
14	50	603.21	**603.21**	575.0	**603.21**	669.0	608.57	575.0	**603.21**	20.41	**603.21**	0.86	**603.21**
15	50	999.82	1007.35	335.0	**999.82**	736.0	**999.82**	289.0	**999.82**	4.61	**999.82**	9.14	**999.82**
16	50	1131.00	1144.39	350.0	**1131.00**	852.0	**1131.00**	246.0	**1131.00**	3.36	**1131.00**	13	**1131.00**
17	75	1038.60	1044.93	2245.0	**1038.60**	1453.0	1047.74	2156.0	**1038.60**	69.38	**1038.60**	9.53	**1038.60**
18	75	1801.40	1831.24	2876.0	**1801.40**	1487.0	1814.11	2469.0	**1801.40**	48.06	**1800.80**	18.92	**1800.80**

Table 10 (*continued*)

Instance	*n*	BKS	Taillard		Gendreau et al.		Wassan & Osman		Choi & Tcha		SMA-U2		Our best
			Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost
19	100	1105.44	1110.96	5833.0	**1105.44**	1681.0	1108.98	2553.0	**1105.44**	182.86	**1105.44**	52.31	**1105.44**
20	100	1530.43	1550.36	3402.0	1541.19	1706.0	1536.24	2910.0	**1530.43**	98.14	1535.12	104.41	1533.29
Av. dev. BKS or av. time			0.773%	2011.1 s	0.088%	1151.3 s	0.462%	1417.3 s	0.000%	53.9 s	0.034%	26.5 s	0.019%
Scaled time for 1.8 GHz				34.7 s		19.9 s		22.7 s		78.0 s		26.5 s	
Nb of BKS retrieved			1		7		2		8		6		6
NB of BKS improved			0		0		0		0		1		1

BKS: best-known solutions taking into account published papers until Choi and Tcha (2007), see corrections done in Section 5.3.
Cost: best cost of 10 runs for Gendreau et al. (1999), of 5 runs otherwise.
Time: duration from original papers, average duration of a run for Taillard (1999), duration of best run otherwise.
Our best: best cost found by our memetic algorithms, using various settings of parameters.
Boldface: BKS retrieved or improved, improvements are also underlined.

Table 11
Best results found on VFMP-FV instances.

Instance	*n*	BKS	Choi & Tcha		SMA-U1		Our best
			Cost	Time	Cost	Time	Cost
3	20	1144.22	**1144.22**	0.25	**1144.22**	0.01	**1144.22**
4	20	6437.33	**6437.33**	0.45	**6437.33**	0.07	**6437.33**
5	20	1322.26	**1322.26**	0.19	**1322.26**	0.02	**1322.26**
6	20	6516.47	**6516.47**	0.41	**6516.47**	0.07	**6516.47**
13	50	2964.65	**2964.65**	3.95	**2964.65**	0.32	**2964.65**
14	50	9126.90	**9126.90**	51.70	**9126.90**	8.90	**9126.90**
15	50	2634.96	**2634.96**	4.36	2635.21	1.04	2635.21
16	50	3168.92	**3168.92**	5.98	3169.14	13.05	3168.95
17	75	2023.61	**2023.61**	68.11	**2004.48**	23.92	**2004.48**
18	75	3147.99	**3147.99**	18.78	3153.16	24.85	3153.16
19	100	8664.29	**8664.29**	905.20	8664.67	163.25	**8662.86**
20	100	4154.49	**4154.49**	53.02	**4154.49**	41.25	**4153.11**
Av. dev. BKS or av. Time			0.000%	92.7 s	−0.063%	23.1 s	−0.068%
Scaled time for 1.8 GHz				134.3 s		23.1 s	
Nb of BKS retrieved			12		7		6
Nb of BKS improved			0		1		3

BKS: best-known solutions taking into account papers until Choi and Tcha (2007).
Cost: best cost of 5 runs. Time: duration of best run.
Our best: best cost found by our memetic algorithms, using various settings of parameters.
Boldface: BKS retrieved or improved, improvements are also underlined.

Table 12
Best results found on HVRP instances.

File	*n*	BKS	Taillard		Tarantilis et al.		Li et al.		SMA-D2		Our best
			Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost
13	50	1517.84	1518.05	473	1519.96	843	**1517.84**	358	**1517.84**	33.2	**1517.84**
14	50	607.53	615.64	575	611.39	387	**607.53**	141	**607.53**	37.6	**607.53**
15	50	1015.29	1016.86	335	**1015.29**	368	**1015.29**	166	**1015.29**	6.6	**1015.29**
16	50	1144.94	1154.05	350	1145.52	341	**1144.94**	188	**1144.94**	7.5	**1144.94**
17	75	1061.96	1071.79	2245	1071.01	363	**1061.96**	216	1065.85	81.5	1064.07
18	75	1823.58	1870.16	2876	1846.35	971	**1823.58**	366	**1823.58**	190.6	**1823.58**
19	100	1117.51	**1117.51**	5833	1123.83	428	1120.34	404	1120.34	177.8	1120.34
20	100	1534.17	1559.77	3402	1556.35	1156	**1534.17**	447	**1534.17**	223.3	**1534.17**
Av dev BKS or av time			0.931	2011.1	0.617	607.1	0.032	285.8	0.077	94.8	0.056
Scaled time 1.8 GHz				34.7		101.7		213.4		94.8	
Nb of BKS retrieved			1		1		7		6		6
Nb of BKS improved			0		0		0		0		0

BKS: best-known solutions as listed in Li et al. (2007).
Time: duration from original papers, for one run except Taillard (best of 5 runs).
Our best: best cost found by our memetic algorithms, using various settings of parameters.
Boldface: BKS retrieved or improved.

Reference

Campos, V., Laguna, M., Marti, R., 2005. Context-independent scatter and tabu search for permutation problems. INFORMS Journal on Computing 17, 111–122.

Choi, E., Tcha, D.-W., 2007. A column generation approach to the heterogeneous fleet vehicle routing. Computers & Operations Research 34, 2080–2095.

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S., 2005. New heuristics for the vehicle routing problem. In: Langevin, A., Riopel, D. (Eds.), Logistic Systems: Design and Optimization. Wiley, Chichester, pp. 279–298.

Cordeau, J.F., Laporte, G., Savelsbergh, M.W.P., Vigo, D., 2007. Vehicle routing. In: Barnhart, C., Laporte, G. (Eds.), Transportation (Handbooks in Operations Research and Management Science vol. 14). Elsevier, Amsterdam, pp. 367–428.

Cormen, T.H., Leiserson, C.L., Rivest, M.L., 1990. Introduction to Algorithms. The MIT Press, Cambridge, MA.

Desrosiers, J., Soumis, F., Desrochers, M., 1988. Routing with time windows by column generation. Networks 14, 545–565.

Dongarra, J.J., 2006. Performance of various computers using standard linear equations software, Technical Report CS-89-85, Computer Science Department, University of Tennessee.

Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., Werneck, R.F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Math. Program. Series A 106, 491–511.

Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, E.D., 1999. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers & Operations Research 26, 1153–1173.

Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing problem. Computers & Operations Research 11, 49–66.

Labadi, N., Lacomme, P., Prins, C., 2006. A memetic algorithm for the heterogeneous fleet VRP. In: Benavent, E., et al. (Eds.), Proceedings of Odysseus 2006. University of Valencia, Spain, pp. 209–213.

Laporte, G., Semet, F., 2002. Classical heuristics for the capacitated VRP. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem. SIAM, Philadephia, pp. 109–128.

Li, F., Golden, B.L., Wasil, E.A., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problems. Computers & Operations Research 34, 2734–2742.

Lima, C.M.R.R., Goldbarg, M.C., Goldbarg, E.F.G., 2004. A memetic algorithm for the heterogeneous fleet vehicle routing problem. Electronic Notes in Discrete Mathematics 18, 171–176.

Moscato, P., 1999. Memetic algorithms: A short introduction. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, New York, pp. 219–234.

Oliver, I.M., Smith, D.J., Holland, J.R.C., 1987. A study of permutation crossover operators on the traveling salesman problem. In: Grefenstette, J.J. (Ed.), Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum, Hillsdale, NJ, pp. 224–230.

Osman, I.H., Salhi, S., 1996. Local search strategies for the mix fleet routing problem. In: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. (Eds.), Modern Heuristic Search Methods. Wiley, Chichester, pp. 131–153.

Prins, C., 2002. Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case. Journal of Mathematical Modelling and Algorithms 1, 135–150.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31, 1985–2002.

Renaud, J., Boctor, F.F., 2002. A sweep-based algorithm for the fleet size and mix vehicle routing problem. European Journal of Operations Research 140, 618–628.

Sörensen, K., Sevaux, M., 2006. MA|PM: Memetic algorithms with population management. Computers & Operations Research 33, 1214–1225.

Taillard, E.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO Operations Research 33 (1), 1–14.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2003. A list-based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Journal of the Operational Research Society 54, 65–71.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 152, 148–158.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM, Philadelphia.

Wassan, N.A., Osman, I.H., 2002. Tabu search variants for the mix fleet vehicle routing problem. Journal of the Operations Research Society 53, 768–782.

Yaman, H., 2006. Formulations and valid inequalities for the heterogeneous vehicle routing problem. Math. Program. Series A 106, 365–390.