

A Memetic Algorithm for the Multi Trip Vehicle Routing Problem

Diego CATTARUZZA, Nabil ABSI, Dominique FEILLET

Ecole des Mines de Saint-Etienne
CMP – Site Georges Charpak
880, Avenue de Mimet
F-13541 Gardanne – France

Thibaut VIDAL

Université de Montréal
CIRRELT – Département d'informatique et de recherche
opérationnelle,
Montréal 7523 – Canada H3C 3J7

November 2012

Working Paper EMSE CMP–SFL 2012/1

We consider the Multi Trip Vehicle Routing Problem, in which a set of geographically scattered customers have to be served by a fleet of vehicles. Each vehicle can perform several trips during the working day. The objective is to minimize the total travel time while respecting temporal and capacity constraints.

The problem is particularly interesting in the city logistics context, where customers are located in city centers. Road and law restrictions favor the use of small capacity vehicles to perform deliveries. This leads to trips much briefer than the working day. A vehicle can then go back to the depot and be re-loaded before starting another service trip.

We propose a hybrid genetic algorithm for the problem. Especially, we introduce a new local search operator based on the combination of standard VRP moves and swaps between trips. Our procedure is compared with those in the literature and it outperforms previous algorithms with respect to average solution quality. Moreover, a new feasible solution and many best known solutions are found.



A Memetic Algorithm for the Multi Trip Vehicle Routing Problem

Diego Cattaruzza¹, Nabil Absi¹, Dominique Feillet¹, and Thibaut Vidal^{2,3}

¹Ecole des Mines de Saint-Etienne, CMP Georges Charpak, F. 13541 Gardanne, France

²CIRRELT, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Canada H3C 3J7

³Institut Charles Delaunay, Université de Technologie de Troyes, 10010 Troyes, France

November 14, 2012

Abstract

We consider the Multi Trip Vehicle Routing Problem, in which a set of geographically scattered customers have to be served by a fleet of vehicles. Each vehicle can perform several trips during the working day. The objective is to minimize the total travel time while respecting temporal and capacity constraints.

The problem is particularly interesting in the city logistics context, where customers are located in city centers. Road and law restrictions favor the use of small capacity vehicles to perform deliveries. This leads to trips much briefer than the working day. A vehicle can then go back to the depot and be re-loaded before starting another service trip.

We propose an hybrid genetic algorithm for the problem. Especially, we introduce a new local search operator based on the combination of standard VRP moves and swaps between trips. Our procedure is compared with those in the literature and it outperforms previous algorithms with respect to average solution quality. Moreover, a new feasible solution and many best known solutions are found.

1 Introduction

The well known Vehicle Routing Problem (VRP) is an \mathcal{NP} -hard combinatorial optimization problem where a set of geographically scattered customers has to be served by a fleet of vehicles. An implicit assumption of the VRP is that each vehicle can perform only one route in the planning horizon. This assumption is not realistic in several practical situations. For the distribution of goods in city centers, for example, small vehicles are generally preferred. Because of this capacity limitation, they daily perform several short tours. This problem is referred to as the Multi Trip VRP (also VRP with multiple use of vehicles, Taillard et al. [30], VRP with multiple trips, Petch and Salhi [21] or VRP with multiple routes, Azi et al. [2]). In the rest of the paper it will be indicated as MTRVP.

The MTRVP is defined on an undirected graph $G = (V, E)$, where $V = \{0, 1, \dots, n\}$ is the set of vertices and $E = \{(i, j) | i, j \in V, i < j\}$ is the set of edges. It is possible to travel from i to j , incurring in a travel time t_{ij} . Vertex 0 represents the depot where a fleet of m identical vehicles with limited capacity Q is based. Vertices $1, \dots, n$ represent the customers to be served, each one having a demand q_i . A

time horizon T_H exists, which establishes the duration of the working day. Overtime is not allowed. It is assumed that Q , q_i and T_H are nonnegative integers.

The MTVRP calls for the determination of a set of routes and an assignment of each route to a vehicle, such that the total travel time is minimized and the following conditions are satisfied:

- (a) each route starts and ends at the depot,
- (b) each customer is visited by exactly one route,
- (c) the sum of the demands of the customers in any route does not exceed Q ,
- (d) the total duration of the routes assigned to the same vehicle does not exceed T_H .

It is also supposed that each customer i could be served by a return trip, i.e., $t_{0i} + t_{i0} \leq T_H$ and $q_i \leq Q$.

Few papers in the literature address the MTVRP and no efficient population-based algorithm were proposed. Our goal is to fill this gap proposing a memetic algorithm able to compete with previous works. Our interest in the MTVRP raises from the MODUM project¹, where mutualized distribution in city centers is explored. The contribution of this paper is threefold: 1) A high-performing memetic algorithm is proposed; 2) An adaptation of the Split procedure (Prins [22]) to segment a chromosome into a MTVRP solution is developed; 3) A new local search (LS) operator, that combines standard VRP moves and re-assignment of trips to vehicles is introduced.

This paper is organized as follows. In Section 2 the literature of the MTVRP is reviewed. Section 3 describes the proposed algorithm. Section 4 details the *Combined* LS. Results are reported in Section 5. Conclusions and perspectives are discussed in Section 6.

2 Literature review

The well known VRP was deeply studied in the last 50 years and many exact and heuristic methods have been proposed in the literature (see Toth and Vigo [32] and Golden et al. [12]). However, exact methods remain limited to problems with restricted size, i.e., less than 100 customers. Moreover, many different variants of the problem are introduced in order to face particular constraints that arise in everyday applications. Despite that, MTVRP has been investigated only in the last two decades and the literature is still scarce.

Fleischmann [10] was the first to address the problem in his working paper in 1990. He proposes a modification of the savings algorithm and uses a bin packing (BP) problem heuristic to assign routes to the vehicles. In Taillard et al. [30], VRP solutions are generated using a tabu search (TS) algorithm with adaptive memory (Taillard [29]). The routes forming the VRP solutions are stored in a list. From that list a subset of routes is selected and a MTVRP solution is constructed using a BP heuristic. A benchmark of instances (constructed from VRP instances) is proposed. It will be used as efficiency comparison for all the authors that have developed a solution method for the MTVRP. Curiously, Taillard et al. [30] provide values only when the algorithm fails in finding a feasible solution, introducing an arbitrary penalization factor $\theta = 2$ for the overtime. Next papers followed the same scheme except Salhi and Petch [27] (Olivera and Viera [20] do not provide exact values, but just a *GAP* measure from a reference value as it will be explained in Section 5). Petch and Salhi [21] propose a multi-phase algorithm with the minimization of the overtime as objective function. A pool of solutions is constructed by the parametrized Yellow's savings algorithm (Yellow [36]). For each solution in the pool, a MTVRP solution is constructed using a BP heuristic. The MTVRP solutions are improved using 2-opt, 3-opt moves, combining routes and reallocating customers. In Salhi and Petch [27], as in Petch and Salhi [21], the maximum overtime is minimized. A genetic algorithm is proposed. In this method a chromosome is a sequence of strictly

¹<http://www-lipn.univ-paris13.fr/modum>

increasing angles, measured with respect to the depot, and dividing the plane into sectors. The customers are then clustered by assigning each one to the sector it occupies. In each cluster, the Clarke and Wright savings heuristic is used to solve a smaller VRP problem. The resulting routes are packed using a BP heuristic. Olivera and Viera [20] use an adaptive memory approach to tackle the MTVRP. A memory M is constructed with different routes that form VRP solutions generated with the sweep algorithm. Each route is labeled with its overtime value and its cost and are sorted using a lexicographic order. New VRP solutions are generated by probabilistically selecting routes in M and improved by a TS algorithm. New VRP solutions are used to update M . From the best VRP solution a MTVRP solution is obtained using a BP heuristic. Recently, Mingozzi et al. [17] propose an exact method for the MTVRP based on two set partitioning-like formulations. 52 instances with up to 120 customers and with a known feasible solution (without overtime) are tackled and in 42 cases the optimal solution is found.

The MTVRP with time widows (MTVRPTW) is addressed as well. Several exact methods are proposed (Azi et al. [2], Hernandez et al. [15]). Instances with 100 customers and 1 vehicle (Azi et al. [2]) and with 50 customers and 4 vehicles (Hernandez et al. [15]) can be solved to optimality.

Different studies facing practical cases envisage to perform several trips during the working day. For example, Brandão and Mercer [4] consider a MTVRPTW and vehicles with different capacities. Moreover, vehicles can be hired from the company in case of need and the access to some customers is restricted to particular vehicles. Drivers' schedule must respect the maximum legal driving time per day. Legal time breaks and unloading times are taken into account. Real instances including 45 to 70 customers and the use of 11 vans and 11 tractors are considered. In their subsequent work, Brandão and Mercer [5] adapt the algorithm to compare the results with those obtained by Taillard et al. [30]. A two phases TS is performed. In the first phase, a solution is allowed to become infeasible regarding travel time constraints, but in the second phase, only feasible solutions are accepted. Insert and swap moves are considered. Battarra et al. [3] consider the MTVRPTW and different commodities that cannot be transported together. The objective is to minimize the number of used vehicles. The problem is decomposed in simpler subproblems, one for each commodity. A set of routes is then generated for each commodity and packed by means of a BP heuristic in order to obtain a solution. Alonso et al. [1] consider the periodic MTVRP. Each customer has to be served up to t times in a planning horizon of t periods. Moreover, not every vehicle can serve all the customers. To each customer is assigned a delivery pattern and it is assigned to a vehicle using GENIUS heuristic (Gendreau et al. [11]). If the insertion violates time or capacity constraints a new route is initialized. Two moves are used to improve the solution: customers are moved from a route to another and different patterns are assigned to a customer. The concept of multi trips is also addressed by Cornillier et al. [8] and Gribkovskaia et al. [13]. The former paper concerns the petrol distribution to gas stations, while the latter proposes a model for the livestock collection.

The idea of multi-trip is found in the context of city logistics as well. For example, Taniguchi and Shimamoto [31] propose a model to evaluate the impact of advanced information system in urban areas and they assume that vehicles are allowed to perform multiple trips per day. Browne et al. [6] present the case of supplies company operating in the City of London. From a micro-consolidation urban center, electrically assisted cargo tricycles and electric vans perform deliveries. Due to the small size of tricycles and electric vans, they perform several trips during each day.

3 A memetic algorithm for the MTVRP

Genetic algorithms (GA) are adaptive methods inspired from the natural evolution of biological organisms. An initial population of individuals (*chromosomes*) evolves through generations until satisfactory criteria of quality, a maximum number of iterations or time limits are reached. New individuals (*children*) are generated from individuals forming the current generation (*parents*) by means of genetic operators (*crossover* and *mutation*). The principles of genetic procedure were firstly formalized by Holland [16] and

have been successfully used in different context (Neri and Cotta [19]). The papers of Prins [22] and Vidal et al. [35] are two examples of efficient GA (the former for the VRP and the latter for the multi depot VRP and the periodic VRP) in the VRP field. In particular, GAs allow for a diversified exploration over the search space due to the management of several solutions at the same time. When LS algorithms are part of the procedure, the GA is commonly called memetic algorithm (MA). For an overview of GAs and MAs the reader is respectively referred to Reeves [25] and Moscato and Cotta [18].

In this section the proposed MA for the MTLVRP is described. It makes use of an adaptation of the Split procedure (Prins [22]) to obtain a MTLVRP solution from *giant tours* (Section 3.2). The population diversity management is inspired by the work of Vidal et al. [35]: for survival, individuals are selected according to their quality and their contribution to the diversification of the population (Section 3.6). A sketch of the method is given in Algorithm 1.

A new advanced feature is embedded in the LS: when a pejorative move is detected, it is tested in combination with a re-assignment of trips. In case of improvement, both the move and the re-assignment are performed (Section 4).

Algorithm 1 Memetic Algorithm outline

- 1: Initialize population (Section 3.5)
 - 2: **while** Termination criteria is not met **do**
 - 3: Select parent chromosomes S_{P_1} and S_{P_2} (Section 3.3)
 - 4: Generate a child S_C (Section 3.3)
 - 5: Educate S_C (Section 3.4)
 - 6: **if** S_C is infeasible **then**
 - 7: Repair S_C (Section 3.4)
 - 8: **end if**
 - 9: Insert S_C in the population
 - 10: **if** Dimension of the population is bigger or equal than $\pi + \mu$ **then**
 - 11: Select survivors (Section 3.6)
 - 12: **end if**
 - 13: **end while**
-

3.1 Solution representation and search space

A chromosome is a sequence (permutation) $S = (S_1, \dots, S_n)$ of n client nodes, without trip delimiters. S can be viewed as a TSP solution that has to be turned in a feasible MTLVRP solution by splitting the chromosome (inserting trip delimiters and assigning trips to vehicles). From that point of view, S is usually called a *giant tour*. From a giant tour S , different MTLVRP solutions can be constructed depending on the way S is split.

During the search phase, overtime and overload are allowed and penalized in the fitness function with factors θ and λ respectively, even though a feasible solution is required.

A procedure *AdSplit* (explained in Section 3.2) is used to get a MTLVRP solution ξ from S . The following notation is introduced: $T_v(\xi)$ and $O_v(\xi) = \max\{0, T_v(\xi) - T_H\}$ are respectively the travel time and the overtime of vehicle v in solution ξ . $L_r(\xi)$ is the load of route r and $r \in v$ indicates that route r is assigned to vehicle v . The fitness $F(S)$ of the chromosome S is the cost of the best solution ξ found by *AdSplit* and it is defined as

$$F(S) = c(\xi) = \sum_{v=1}^m T_v(\xi) + \theta \sum_{v=1}^m O_v(\xi) + \lambda \sum_{v=1}^m \sum_{r \in v} \max\{0, L_r(\xi) - Q\} \quad (1)$$

When confusion cannot arise, solution ξ will be omitted in the notation. The chromosome S is called *feasible (infeasible)* if *AdSplit* obtains, from S , a feasible (infeasible) solution ξ .

3.2 A Split algorithm for the Multi Trip problems

3.2.1 Auxiliary graph construction

The splitting procedure proposed here, indicated with *AdSplit*, is an adaptation of the procedure proposed by Prins in [22]. It works on an auxiliary graph $H = (V', A')$. V' contains $n + 1$ nodes indexed from 0 to n . Arc (i, j) , $i < j$, represents a trip serving customers from S_{i+1} to S_j in the order they are in S . With each arc (i, j) , is associated a cost c_{ij} defined as

$$c_{ij} = \tau_{ij} + \theta \max\{0, \tau_{ij} - T_H\} + \lambda \max\{0, l_{ij} - Q\} \quad (2)$$

where τ_{ij} and l_{ij} represent respectively the trip travelling time and the sum of customer's requests served during the trip.

A simple example with five customers is given in Figures 1–2. $S = (1, 2, 3, 4, 5)$, $T_H = 45$, $Q = 50$, $\theta = \lambda = 2$ and the demand of each customer is given between brackets. For example, arc $(1, 5)$ in Figure 2 represents the trip serving customers from 2 to 5. $\tau_{15} = 116$, $l_{15} = 76$. The arc cost is then $c_{15} = 116 + 2 \cdot (116 - 45) + 2 \cdot (76 - 50) = 310$.

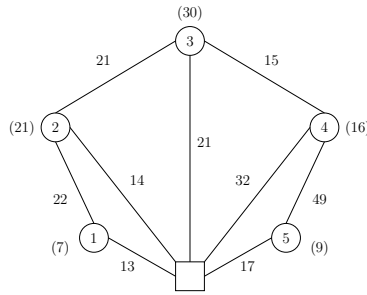


Figure 1: Example with 5 customers: demands in brackets, $T_H = 45$, $Q = 50$, $\theta = \lambda = 2$

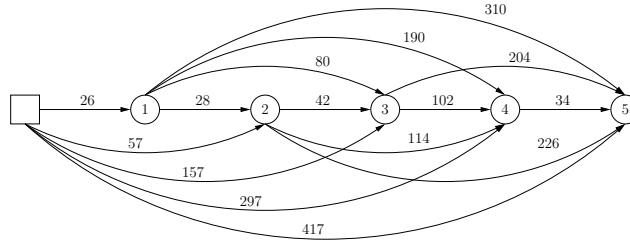


Figure 2: Auxiliary graph (to each arc (i, j) , cost c_{ij} is assigned as defined in Equation 2)

Once H is computed, paths basically represent set of trips that can be assigned to vehicles. In the VRP context, an optimal splitting is equivalent to a shortest path (SP) in H each arc representing a route which is assigned to a vehicle. Since H is acyclic, Bellman's algorithm can be used to find the SP in $O(n^2)$. In the MTRVP context, more than one trip can be assigned to the same vehicle. The procedure proposed in Prins [22] cannot be directly used and is modified as explained in Section 3.2.2.

3.2.2 Assignment procedure

The assignment procedure both selects and assigns trips to vehicles. It consists of two phases. In the first phase, the SP in H is computed. In the second phase, trips of the SP are assigned to vehicles by means of a labelling algorithm. The labelling algorithm works as follows.

Starting from node 0, labels are progressively extended along the graph defined by SP. Each label \mathcal{L} has $m + 3$ fields: the first m fields store vehicle travel times in decreasing order, the $(m + 1)^{\text{th}}$ field

memorizes the total load infeasibility, the $(m + 2)^{\text{th}}$ the predecessor node, and the last field keeps the cost of the partial solution evaluated using Equation 1 and equivalent to the cost $c(\mathcal{L})$ of label \mathcal{L} . When extending a label, m new labels are constructed, one for each possible allocation of the new trip to a vehicle. When node n is reached, the label \mathcal{L} with minimum cost $c(\mathcal{L})$ associated with node n is selected and the related solution is constructed.

Dominated labels, accordingly with the following dominance rule, are discarded: let \mathcal{L}^1 and \mathcal{L}^2 be two labels associated with the same node i . \mathcal{L}^1 dominates \mathcal{L}^2 if and only if

$$c(\mathcal{L}^1) + \theta \sum_{j=1}^m \delta_j(\mathcal{L}^1, \mathcal{L}^2) \leq c(\mathcal{L}^2) \quad (3)$$

where $c(\mathcal{L})$ is the cost associated with label \mathcal{L} ,

$$\delta_j(\mathcal{L}^1, \mathcal{L}^2) = \max \left\{ 0, \min \{ T_H, T_j(\mathcal{L}^1) \} - \min \{ T_H, T_j(\mathcal{L}^2) \} \right\}$$

and $T_j(\mathcal{L})$ is the (partial) travel time of vehicle j associated to label \mathcal{L} . Roughly speaking, given two labels \mathcal{L}^1 and \mathcal{L}^2 , extending \mathcal{L}^1 is penalized as much as possible while it is not extending \mathcal{L}^2 in the same way. If Inequality 3 holds, \mathcal{L}^2 cannot be extended in a better way than \mathcal{L}^1 , and it is eliminated.

Note that this approach provides the optimal assignment of trips in SP, but is suboptimal with regard to the decomposition of S , as illustrated in Figures 3 and 4.

Applying the procedure on the complete graph H , the label that minimizes Equation 1 at node n would correspond to the best decomposition of S in the MTVRP context. One could however expect that a huge number of labels would need to be treated, which do not appear to be viable in the MA context.

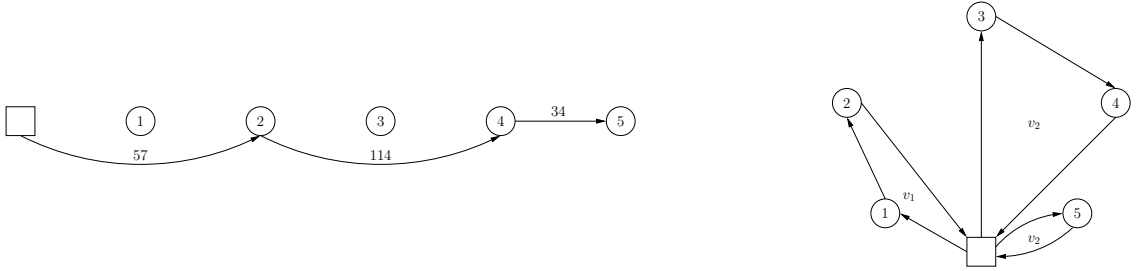


Figure 3: Best MTVRP solution for S ($F(S) = 273$)

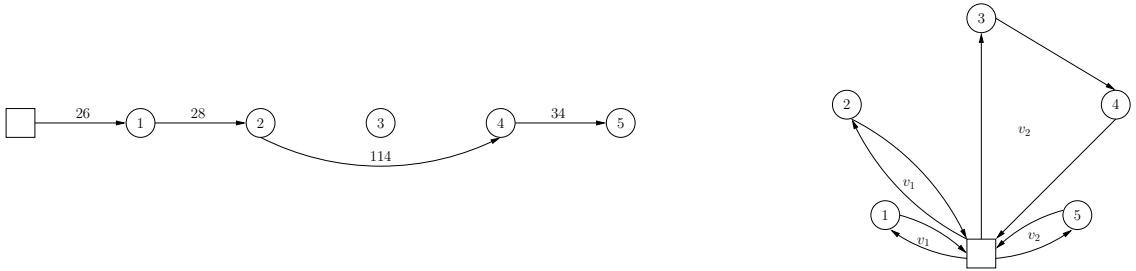


Figure 4: MTVRP solution obtained from arcs forming the shortest path ($F(S) = 288$): shortest path could not lead to the best solution

3.2.3 Improving the Split procedure

Arc (i, j) in the auxiliary graph H represents the trip serving customers $(S_{i+1}, S_{i+2}, \dots, S_j)$ in the order they appear in the giant tour S . Visiting customers in a different order can lead to a trip with a smaller cost (Figure 5). As proposed by Prins et al. [24], each rotation (circular left shift) can be considered and evaluated in constant time. For example, a one-position rotation corresponds to the trip

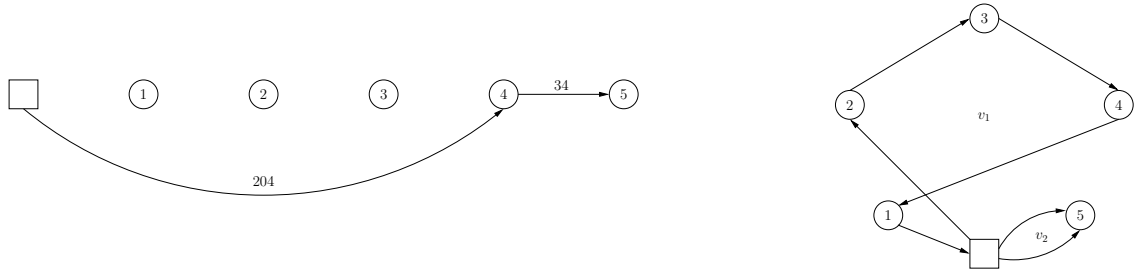


Figure 5: MTVRP solution obtained considering rotations of customers in the same trip ($F(S) = 238$)

$(0, S_{i+2}, \dots, S_j, S_{i+1}, 0)$. Then, given an arc, all the possible rotations are considered looking for the best starting point of the trip without introducing any computational burden (see Prins et al. [24] for a detailed explanation).

A pseudo-code sketch of the *AdSplit* procedure is proposed in Algorithm 2. Procedure *SP_best_in()* computes the shortest path on graph H , taking into consideration the best rotation for each arc. With each node i , it associates its successor $succ_i$, the travelling time and load of the trip represented by $(i, succ_i)$. These values are obtained when needed by procedures *get_successor(i)*, *get_best_in_time(i)* and *get_load(i)*. \mathcal{L}_k indicates the k^{th} field of label \mathcal{L} , while \mathcal{L}_{Inf} , \mathcal{L}_{pred} and \mathcal{L}_c refer respectively to the $(m+1)^{\text{th}}$, $(m+2)^{\text{th}}$ and $(m+3)^{\text{th}}$ label fields. *sort*(\mathcal{L}) sorts the first m fields in decreasing order. If \mathcal{L} is dominated by a label in $ListLabel_i$, *is_dominated*($ListLabel_i, \mathcal{L}$) returns **true**, otherwise it returns **false**. Labels in $ListLabel_i$ dominated by the new inserted label \mathcal{L} are eliminated from the list by *eliminate_dominated_labels*($ListLabel_i, \mathcal{L}$).

3.3 Crossover

The classic *OX* operator is used. Figure 6 shows how the *OX* works. Firstly, two cutting points have to be chosen in the first parent S_{P_1} . In the example they are $i = 4$ and $j = 7$. Indicating with $S_{C_1} = OX(S_{P_1}, S_{P_2})$ the first child, $S_{C_1}(k) = S_{P_1}(k)$ for $k = i, \dots, j$. Then, S_{P_2} is circularly swept from $S_{P_2}(j+1)$ onward inserting in S_{C_1} the missing nodes. By inverting the roles between S_{P_1} and S_{P_2} , we obtain the second child $S_{C_2} = OX(S_{P_2}, S_{P_1})$.

			i=4				j=7			
			↓				↓			
S_{P_1} :	2	6	4	7	8	5	10	9	3	1
S_{P_2} :	3	7	9	2	4	10	1	6	5	8
S_{C_1} :	2	4	1	7	8	5	10	6	3	9
S_{C_2} :	7	8	5	2	4	10	1	9	3	6

Figure 6: OX operator

Parents S_{P_1} and S_{P_2} are selected with the classic *binary tournament* method: two chromosomes are randomly drawn from the population and the one with the lower fitness is selected. The procedure is repeated twice, once for the selection of each parent. The child that has to be inserted in the population is randomly selected between children $S_{C_1} = OX(S_{P_1}, S_{P_2})$ and $S_{C_2} = OX(S_{P_2}, S_{P_1})$.

Algorithm 2 *AdSplit*

```
1:  $SP\_best\_in()$ 
2: for  $i = 0$  to  $n$  do
3:    $LabelList_i = \emptyset$ 
4: end for
5:  $LabelList_0 \leftarrow (\overbrace{0, \dots, 0}^m, 0, 0, 0)$ 
6:  $current = 0$ 
7: while  $current < n$  do
8:    $succ = get\_successor(current)$ 
9:    $load = get\_load(current)$ ;
10:   $time = get\_best\_in\_time(current)$ 
11:  for all  $\mathcal{L} \in LabelList_{current}$  do
12:    for  $k = 1 \rightarrow m$  do
13:       $\mathcal{L}^* = \mathcal{L}$ 
14:       $\mathcal{L}_k^* = \mathcal{L}_k + time$ 
15:       $sort(\mathcal{L})$ 
16:       $\mathcal{L}_{Inf}^* = \mathcal{L}_{Inf} + \lambda \cdot \max\{load - Q, 0\}$ 
17:       $\mathcal{L}_c^* = \mathcal{L}_c + time + \theta \cdot \max\{time - T_H, 0\} + \lambda \cdot \max\{load - Q, 0\}$ 
18:       $\mathcal{L}_{pred}^* = current$ 
19:      if not  $is\_dominated(ListLabel_{succ}, \mathcal{L}^*)$  then
20:         $ListLabel_{succ} \leftarrow \mathcal{L}^*$ 
21:         $eliminate\_dominated\_labels(ListLabel_{succ}, \mathcal{L}^*)$ 
22:      end if
23:    end for
24:  end for
25:   $current = succ$ 
26: end while
```

3.4 Local search - education and repair procedures

After crossover, the obtained child is evaluated by means of *AdSplit* procedure, and *educated* applying LS procedure with a probability p_{LS} trying to improve its quality. LS is usually used in literature as mutation operator in order to obtain a high-performance hybrid GA.

The operators listed in the following are used. Let u and z be two nodes and t and x be their respective successors (that could be the depot as well). $R(u)$ indicates the route visiting customer u . The following simple types of moves are tested

M1 If u is a client node, remove u and insert it after z ;

M2 If u and t are clients, remove them and insert u and t after z ;

M3 If u and t are clients, remove them and insert t and u after z ;

M4 If u and z are clients, swap u and z ;

M5 If u , t and z are clients, swap u and t with z ;

M6 If u , t , z and x are clients, swap u and t with z and x ;

M7 If $R(u) = R(z)$, replace (u, t) and (z, x) by (u, z) and (t, x) ;

M8 If $R(u) \neq R(z)$, replace (u, t) and (z, x) by (u, z) and (t, x) ;

M9 If $R(u) \neq R(z)$, replace (u, t) and (z, x) by (u, x) and (t, x) ;

M10 If $R(u) = R(z)$, create another route with all customers from u to z (of from z to u if z come before u) and put it in a randomly drawn vehicle.

The nodes can belong to the same route or to different routes. Routes can either belong to the same vehicle or to different vehicles. Moves M1–M3 correspond to *insertion* moves, moves M4–M6 to *swaps*, move M7 is the well known *2-opt* and moves M8, M9 are usually called *2-opt**.

Moves M1–M9 are those used in Prins [22]. If $u = z$ in M10, a new route with just customer u is created.

At the beginning of the LS with each type of move M_i , $i = 1, \dots, 10$ is associated a weight $w_i = w$ and the status *active*. At each iteration the LS procedure probabilistically selects a move among the active moves. The probability of move M_i to be chosen is w_i/W where $W = \sum_{i=1}^{10} w_i$. The selected move M_i is evaluated and the first improvement criteria is adopted. If the move fails, i.e., the current solution is a local optima in the neighbourhood defined by M_i , M_i becomes *inactive* and cannot be selected anymore until another move succeeds. The LS terminates when all the moves are inactive, i.e., a local optima in the neighbourhood defined by M1–M10 is reached.

After a fixed number of iterations ω (arbitrarily fixed to 100), the weights are updated accordingly to the number of successes. Precisely, $w_i = w_i + \frac{\text{success}_i}{\text{attempts}_i}$, where success_i and attempts_i indicate respectively the times move M_i succeeded and was performed (attempts_i is usually not the same for all moves due to probabilistic selection). W is updated accordingly. Weights w_i can be viewed as a short-term memory, i.e., a move that historically successes more will have a higher probability to be chosen.

To speed up LS, granular search is implemented as proposed by Toth and Vigo [33]: a move is considered only when z is one of the n_{closest} closest customers of u (*filtering rule*).

Each time a solution ξ is obtained from chromosome S by means of *AdSplit* it is stored in four different n -size vectors that memorize in i^{th} position the predecessor, the successor, the vehicle and the route of customer i . The travel time of each vehicle and the load of each route are stored as well. In this way, moves M1–M9 are evaluated in constant time, while M10 in $O(n)$. Then, given a solution ξ and defining

its neighbourhood $N(\xi)$ by the set of moves M1–M10, it can be completely explored in $O(n^3)$ time (more precisely, in $O(n^2 \cdot n_{closest})$ with the usage of the filtering rule), although the neighbourhood defined by M1–M9 requires $O(n^2)$ operations to be explored.

After LS is applied, the educated chromosome can be either feasible or infeasible. In the latter case the *repair* procedure is applied with a probability p_{rep} . It consists in applying again LS with λ (load infeasibility penalization parameter) and/or θ temporarily multiplied by 10, regarding the nature of the infeasibility. If a feasible chromosome is obtained, it is inserted in the population, otherwise λ and/or θ are (temporarily) multiplied again by 10 and LS reapplied. The original chromosome is not discarded even if the repaired chromosome is feasible (Vidal et al. [35]). All the chromosomes obtained during LS and repair procedure are as well inserted in the population.

3.5 Population structure and initialization

An ordered population Π of π chromosomes is kept. A key value k_S is associated with each chromosome S and the population is sorted regarding the key value. k_S corresponds to the fitness $F(S)$ of S multiplied by a penalization factor P . $P = 1$ if S is feasible, $P = 1.5$ if S is time-infeasible, $P = 2$ if S is load-infeasible, $P = 3$ if S is both load and time-infeasible. This is done in order to ensure the best feasible solution found so far corresponds to the chromosome in the first position of the population (infeasible individuals can cost less than the best feasible one) and in general to keep good quality individuals at the top of Π . Moreover, it allows to manage both feasible and infeasible chromosomes in the same population, differently from what is done, for example, in Vidal et al. [35], where the population is divided in two subpopulations, one for feasible and the other for infeasible chromosomes.

The initial population is formed of π random generated chromosomes evaluated with the *AdSplit* procedure and improved applying LS.

3.6 Survivor strategy

When the population reaches a maximum dimension, i.e., $\pi + \mu$, a survivor selection is performed as proposed by Vidal et al. [34], [35]. Survivor chromosomes are selected based on quality, i.e., on fitness $F(S)$, and their diversity contribution $f(S)$ defined as the average distance between S and its n_c closest neighbours in Π (forming set N_c) as follows:

$$f(S) = \frac{1}{n_c} \sum_{S_1 \in N_c} D(S, S_1), \quad (4)$$

where $D(\cdot, \cdot)$ is the *broken pair* distance that is the number of pairs of adjacent customers in S that are broken in S_1 (Prins [23]). $D(\cdot, \cdot)$ gives a measure on the amount of common arcs between S and S_1 . A biased fitness $bF(\cdot)$ is calculated for each chromosome as follows:

$$bF(S) = r_F(S) + \left(1 - \frac{n_e}{|\Pi|}\right) r_f(S) \quad (5)$$

where $r_F(S)$ and $r_f(S)$ are the ranks of chromosome S calculated based on fitness F and function f defined in Equation 4 respectively, and n_e is a parameter that ensures elitism properties during selection (see Vidal et al. [35] for a formal proof).

4 Combined Local Search

To optimize the packing of routes into vehicles, we introduce the possibility of a re-pack of trips along with a *pejorative* move M among M1–M10 introduced in Section 3.4. By pejorative move, we mean a

move that does not decrease the solution cost. The swap between trips (*Swp*) in different vehicles is used as re-assignment procedure.

To understand the idea of the Combined LS (CLS) let focus on Figures 7–9. The example involves three vehicles with up to three routes each and $T_H = 100$ that is violated by the third vehicle (Figure 7). Move M involves routes r_2 and r_3 of vehicles v_1 and v_2 respectively and it leads to the configuration shown in Figure 8 with an increasing in the solution cost of 5 units (due to the increasing in routing cost). Since M is pejorative, it would be discarded by the LS procedure. However, with a different assignment of trips to vehicles, an improvement can be obtained. In the particular case, it consists of swapping r_2 in v_1 with r_2 in v_3 (Figure 9).

	r_1	r_2	r_3	T_v	θO_v
v_1	60	30		90	-
v_2	30	30	30	90	-
v_3	45	30	30	105	10
				cost:	295

Figure 7: Initial configuration

	r_1	r_2	r_3	T_v	θO_v
v_1	60	25		85	-
v_2	30	30	40	100	-
v_3	45	30	30	105	10
				cost:	300

Figure 8: Pejorative move. In bold trips involved in M

	r_1	r_2	r_3	T_v	θO_v
v_1	60	30		90	-
v_2	30	30	40	100	-
v_3	45	25	30	100	-
				cost:	290

Figure 9: After *Swp*. In bold trips involved in *Swp*

The goal of the CLS is to detect when the combination of moves M1–M10 along with a swap of two trips leads to a better solution and, in that case, to perform both the move *and* the swap.

For the sake of computing time, the main issue here is to avoid evaluating every possible combination of moves with swaps (indicated with $M+Swp$). We propose to limit the evaluations of $M+Swp$ according to the following rule:

Rule 1 (\mathcal{R}_1). *The evaluations of $M+Swp$ is limited to those that would improve the solution even if the assignment of routes to vehicles is optimal before $M+Swp$ is applied.*

Using the subsequent propositions, it is then possible to limit heavily the size of the neighborhood explored with $M+Swp$.

In the following, we will note respectively ξ , ξ^M , ξ^{M+Swp} the current solution, the solution after applying move M and the solution after performing *Swp* as shown in Figure 10. A vehicle without (with) overtime will be called *feasible* (*infeasible*).

$$\xi \xrightarrow{M} \xi^M \xrightarrow{Swp} \xi^{M+Swp}$$

Figure 10: Notation

It is noteworthy that *Swp* can modify overtime, but does not affect the traveling time and the load infeasibility of the solution. We start discussing the choice of swaps.

Proposition 1. *Under rule \mathcal{R}_1 , we can restrict the choice of the Swp as follows:*

1. Swp involves (trips in) two different vehicles v_1 and v_2 ,
2. exactly one vehicle between v_1 and v_2 is feasible,
3. at least one vehicle between v_1 and v_2 has to have been involved in M .

Proof. The following notation is introduced. ΔO refers to a difference in the overtime of the solution induced by move M or *Swp*. In particular $\Delta O(\xi^M) = O(\xi^M) - O(\xi)$ and $\Delta O(\xi^{M+Swp}) = O(\xi^{M+Swp}) - O(\xi^M)$.

We can notice that swapping trips belonging to the same vehicle v cannot lead to any improvement: T_v is not reduced, then O_v is not reduced neither (that proves 1). Let consider two trips belonging to two different vehicles v_1 and v_2 . For ease of notation, we note r_i the trip that belongs to v_i ($r_i \in v_i$) and τ_i , instead of τ_{r_i} , the travel time of trip r_i . Let T_1 (resp., O_1) and T_2 (resp., O_2) be the respective travel times (resp., overtimes) of the two vehicles. If point 2 does not hold, we will prove that swapping r_1 with r_2 cannot improve the solution.

Let suppose both vehicles are feasible or both are infeasible, i.e., $T_i(\xi^M) \leq T_H$ or $T_i(\xi^M) > T_H$, $i = 1, 2$. We consider the two cases separately.

- a. $T_1(\xi^M) \leq T_H$ and $T_2(\xi^M) \leq T_H$. $O_1(\xi^M) = O_2(\xi^M) = 0$. No improvement can be carried out with *Swp*.
- b. $T_1(\xi^M) > T_H$ and $T_2(\xi^M) > T_H$. We consider without loss of generality $\tau_2(\xi^M) \leq \tau_1(\xi^M)$. With *Swp*, overtime of vehicle v_1 decreases; $\Delta O_1(\xi^{M+Swp}) = \max\{\tau_2(\xi^M) - \tau_1(\xi^M), T_H - T_1(\xi^M)\}$. Overtime of vehicle v_2 increases; $\Delta O_2(\xi^{M+Swp}) = \tau_1(\xi^M) - \tau_2(\xi^M)$. Then $\Delta O(\xi^{M+Swp}) = \Delta O_1(\xi^{M+Swp}) + \Delta O_2(\xi^{M+Swp}) \geq \tau_2(\xi^M) - \tau_1(\xi^M) + \tau_1(\xi^M) - \tau_2(\xi^M)$, that is, $\Delta O(\xi^{M+Swp}) \geq 0$.

This proves point 2. Point 3 directly follows from the rule \mathcal{R}_1 . Let us suppose v_1 and v_2 are not involved in M . Then, $\tau_k(\xi) = \tau_k(\xi^M)$ for all $r_k \in v_i$, $i = 1, 2$. If an improvement is obtained by *Swp*, the same improvement could have been obtained applying *Swp* before M (that does not modify trips involved in *Swp*). This means that when the initial assignment of trips to vehicles is optimal, no improvement can be carried out. □

Let us now move the discussion to the choice of the move M to be tested along with a *Swp*. We introduce the following proposition.

Proposition 2. *Under Proposition 1 we can restrict the choice of moves involved along with a swap to those such that*

$$\tau_r(\xi^M) < \tau_r(\xi) \quad \text{for at least one route } r \text{ involved in } M. \quad (\mathcal{C}_1)$$

Proof. We suppose the assignment of trips to vehicles is optimal before M is applied. We will show that when \mathcal{C}_1 does not hold, $M + \text{Swp}$ cannot improve the solution cost. Applying \mathcal{R}_1 , such moves can be discarded. We indicate respectively with R and R^M the set of trips that form ξ and ξ^M . Without loss of generality we can suppose $|R| = |R^M|$ (if M creates a new route, an empty route could be added in R). Let us indicate with r a trip in R and with r_M the corresponding trip in R^M after M have been applied. The following considerations are valid.

- ① The cost of the solution ξ^{M+Swp} is greater than or equal to the cost of the solution obtained by optimally assigning trips in R^M to vehicles. We indicate such solution with $\xi_{R^M}^*$ and its cost with $c_{R^M}^*$;

- ② Let $\tilde{\xi}$ be the solution constructed by assigning trips in R as follows: r is assigned to vehicle v if and only if the corresponding r_M is assigned to vehicle v in $\xi_{R^M}^*$. We note \tilde{c} the cost of such solution. Since $\tau_{r_M} \geq \tau_r$ for all $r \in R$ (\mathcal{C}_1 does not hold), $c_{R^M}^* \geq \tilde{c}$ is verified.
- ③ We note c_R^* the cost of the solution obtained by optimally assigning trips in R to vehicles. Then, it holds $\tilde{c} \geq c_R^*$.
- ④ We have assumed the initial assignment of trips to vehicles to be optimal. Then, $c_R^* = c(\xi)$.

Concluding, the following holds

$$c(\xi^{M+Swap}) \stackrel{\textcircled{1}}{\geq} c_{R^M}^* \stackrel{\textcircled{2}}{\geq} \tilde{c} \stackrel{\textcircled{3}}{\geq} c_R^* \stackrel{\textcircled{4}}{=} c(\xi),$$

namely, $M + Swap$ cannot improve the solution ξ . □

An algorithm sketch of the procedure is given in Algorithm 3. *Detect_Trips_To_Swap*(v_1, v_2) is a

Algorithm 3 Combined LS

```

1: evaluate move  $M$ 
2: if  $M$  improves the solution then
3:   accept  $M$ 
4: else
5:   if  $\mathcal{C}_1$  then
6:     for all  $v_1$  involved in  $M$  do
7:       for all  $v_2 \neq v_1$  do
8:         if  $(T_1(\xi^M) < T_H \wedge T_2(\xi^M) > T_H) \vee (T_1(\xi^M) > T_H \wedge T_2(\xi^M) < T_H)$  then
9:            $(r_1, r_2, tripDetected) = Detect\_Trips\_To\_Swap(v_1, v_2)$ 
10:          if  $tripDetected$  then
11:             $perform(M)$ 
12:             $swap(v_1, v_2, r_1, r_2)$ 
13:          end if
14:        end if
15:      end for
16:    end for
17:  end if
18: end if

```

function that tests swaps between trips in vehicles v_1 and v_2 . If it finds a pair of trips r_1, r_2 that improves the solution if swapped, it returns them and sets *tripDetected* to TRUE. Otherwise *tripDetected* is set to FALSE. Function *perform*(M) performs move M while *swap*(v_1, v_2, r_1, r_2) swaps trips r_1, r_2 .

5 Computational results

This section reports the computational results obtained with the proposed method. The algorithm is coded in C++, compiled with Visual Studio 2008 and run on a Intel Xeon 2.80 Ghz processor. It is tested on classical instances in the MTRVP literature. These instances were introduced by Taillard et al. [30] and are constructed from the instances 1–5 and 11–12 proposed in Christofides et al. [7] (that will be denoted CMT1–CMT5 and CMT11–CMT12 in the following) and instances 11–12 proposed in Fisher [9] (F11–F12) for the VRP. For each VRP instance, instances for MTRVP are constructed with different values of m and two values of T_H , given by $T_H^1 = \left\lceil \frac{1.05z^*}{m} \right\rceil$ and $T_H^2 = \left\lceil \frac{1.1z^*}{m} \right\rceil$ where z^* is the solution cost of

Instance	n	Q	z^*
CMT1	50	160	524.61
CMT2	75	140	835.26
CMT3	100	200	826.14
CMT4	150	200	1028.42
CMT5	199	200	1291.44
CMT11	120	200	1042.11
CMT12	100	200	819.56
F11	71	30000	241.97
F12	134	2210	1162.92

Table 1: Instances' details

the original CVRP instances found by Rochat [26] and $[x]$ represents the closest integer to x (see Table 1). There are, in total, 104 different instances. For 42 of them, the optimal value is known and is provided by Mingozzi et al. [17]. We classify them in a first group denoted G1. For the remaining 62 instances, 56 have a known feasible solution (they will form a second group G2). The six remaining instances form the third group G3. These instances are not solved yet. These groups of the instances set will be used during the presentation of the computational results. When it is necessary to indicate a specific instance, the notation $N_T_H^i_m$, will be used, where N stands for the original VRP instance name and $i = 1, 2$ for the horizon length.

5.1 Parameter settings

5.1.1 Overtime and overload penalization parameters

The overtime penalization parameter θ is set to 2 and it is kept fixed during all the search. That is done because the value $\theta = 2$ is used in literature to penalize overtime when a feasible solution is not found.

The overload penalization parameter λ is set to \bar{d}/\bar{q} , where \bar{d} represents the average distance among customers and \bar{q} the average demand of customers. The value of λ is kept fixed during the search. Different dynamic adaptation schemes were tested, but no visible improvements were obtained.

5.1.2 Parameter tuning

The procedure requires the setting of some parameters among values that have to be chosen in sensible ranges. To set the parameters involved in our algorithm, a *tuning method* is used. Roughly speaking, a tuning method is a procedure whose search space is $P^1 \times \dots \times P^{n_p}$, where P^i is the domain of parameter i and looks for the solution with the best *utility*, that is a measure of the algorithm's efficiency on a given parameter vector (Smit and Eiben [28]). In particular, the Evolutionary Strategy with Covariance Matrix Adaptation proposed by Hansen and Ostermeier [14] is used. The tuning algorithm is run on a limited set of instances formed by CMT1_ T_H^2 _4, CMT2_ T_H^1 _6, CMT3_ T_H^1 _6, CMT4_ T_H^1 _8, CMT5_ T_H^1 _9, CMT11_ T_H^1 _4 to determine the values of parameters listed in Table 2. Instances with a large number of vehicles were selected since they are more difficult to solve. Other parameters are fixed a priori: the probability of educate a new chromosome is $p_{LS} = 1$ and the probability to repair an infeasible chromosome is $p_{rep} = 0.5$ as in Vidal et al. [34]. The adopted survivor strategy (Section 3.6) allows for the use of LS to educate each chromosome without premature convergence of the population. That is in particular due to the fact that survivor chromosomes are selected based on their contribution to the diversification of the population as well as their fitness value.

Parameter		Range	Final value
Π	Dimension of population	[1, 100]	9
μ	Children generated at each generation	[1, 100]	32
n^e	Proportion of elite individuals $n_e = n^e \times \Pi$ (Eq. 4)	[0.1, 1]	0.2
n^c	Proportion of close individuals $n_c = n^c \times \Pi$ (Eq. 5)	[0.1, 1]	0.35
h	Granularity threshold in LS $n_{closest} = h \times n$	[0.2, 1]	0.45

Table 2: Parameter Tuning

5.2 Discussion

A fair and comprehensive comparison with previous works is quite difficult to carry out since (as already mentioned) complete and precise values are reported only by Salhi and Petch [27].

Olivera and Veira [20], report detailed results as well, but with some imprecision. Indeed, these authors provide gaps to values z^* (see Table 1), which cannot be precisely converted into solution costs due to truncation.

Notation reported in Table 3 will be used in the following. In all tables, the first three columns indicate respectively the name, the number of vehicles and the time horizon of the instances.

Symbol	Meaning
TLG	results from Taillard, Laporte and Gendreau [30]
BM	results from Brandão and Mercer [5]
SP	results from Salhi and Petch [27]
OV	results from Olivera and Viera [20]
AAB	results from Alonso, Alvarez and Beasley [1]
MRT	results from Mingozzi, Roberti and Toth [17]
MAMTVRP-F	results from our MA stopping at the first feasible found solution
MAMTVRP	results from MA without the usage of CLS
MAMTVRP+CLS	results from our MA with the usage of CLS
Best	Best value over five runs
Av	Average value over five runs
Worst	Worst value over five runs
StDv	Standard deviation over five runs
#fs	Number of runs ended with a feasible solution
#opt	Number of runs ended with an optimal solution
✓	a feasible solution is found
×	a feasible solution is not found
✗	the instance is not considered

Table 3: Notation for computational results

The results are reported as follows. In Section 5.2.1, the ability of the algorithm to find feasible solutions is tested. The procedure terminates as soon as a feasible solution is obtained. In Section 5.2.2 two variants of the algorithm, with or without CLS, are evaluated and complete and detailed results are reported. Both versions stop after a fixed number of iterations or when an optimal solution is obtained. Separate comparison with the results obtained by Olivera and Viera [20] is discussed in Section 5.2.3. Finally, computational times comparison is discussed in Section 5.2.4.

5.2.1 Feasibility check algorithm

The procedure is first run five times over all instances to measure its capability to obtain feasible solutions: it stops as soon as a feasible solution is found. It is indicated as MAMTVRP-F. The efficiency of the algorithm is measured on the time needed to find a feasible solution without considering its value, following the implicit idea of the paper by Taillard et al. [30]. Results are reported on Tables 4 and 5.

Instance			Algorithm						
Name	m	T_H	TLG	BM	SP	OV	AAB	MAMTVRP-F	#fs
CMT1	1	551	✓	✓	✓	✓	✓	✓	5
	2	275	✓	✓	×	✓	✓	✓	5
	1	577	✓	✓	✓	✓	✓	✓	5
	2	289	✓	✓	✓	✓	✓	✓	5
	4	144	✓	✓	✓	✓	✓	✓	5
CMT2	1	877	✓	✓	✓	✓	✓	✓	5
	2	439	✓	✓	✓	✓	✓	✓	5
	3	292	✓	✓	×	✓	✓	✓	5
	4	219	✓	✓	✓	✓	✓	✓	5
	5	175	✓	✓	×	✓	✓	✓	5
	1	919	✓	✓	✓	✓	✓	✓	5
	2	459	✓	✓	✓	✓	✓	✓	5
	3	306	✓	✓	✓	✓	✓	✓	5
	4	230	✓	✓	✓	✓	✓	✓	5
	5	184	✓	✓	✓	✓	✓	✓	5
	6	153	✓	✓	×	✓	✓	✓	5
CMT3	1	867	✓	✓	✓	✓	✓	✓	5
	2	434	✓	✓	✓	✓	✓	✓	5
	3	289	✓	✓	×	✓	✓	✓	5
	1	909	✓	✓	✓	✓	✓	✓	5
	2	454	✓	✓	✓	✓	✓	✓	5
	3	303	✓	✓	✓	✓	✓	✓	5
	4	227	✓	✓	✓	✓	✓	✓	5
CMT11	1	1094	✓	✓	✓	✓	✓	✓	5
	2	547	✓	✓	×	✓	✓	✓	5
	3	365	✓	✓	×	✓	✓	✓	5
	5	219	✓	✓	×	✓	✓	✓	5
	1	1146	✓	✓	✓	✓	✓	✓	5
	2	573	✓	✓	✓	✓	✓	✓	5
	3	382	✓	✓	✓	✓	✓	✓	5
	4	287	✓	✓	×	✓	✓	✓	5
	5	229	✓	✓	✓	✓	✓	✓	5
	CMT12	1	861	✓	✓	✓	✓	✓	✓
2		430	✓	✓	✓	✓	✓	✓	5
3		287	✓	✓	✓	✓	✓	✓	5
4		215	✓	✓	✓	✓	✓	✓	5
1		902	✓	✓	✓	✓	✓	✓	5
2		451	✓	✓	✓	✓	✓	✓	5
3		301	✓	✓	✓	✓	✓	✓	5
4		225	✓	✓	✓	✓	✓	✓	5
5		180	✓	✓	✓	✓	✓	✓	5
6		150	✓	✓	✓	✓	✓	✓	5
# problems solved			42	42	34	42	42	42	

Table 4: Feasibility check on the 42 instances in G1

The algorithm is able to find a feasible solution in at least one run on all instances from groups G1 and G2. Better, feasible solutions are always found on G1 and for 51 instances out of 56 on G2. In general, on all the 490 runs, feasible solution are obtained 474 times, denoting high efficiency of the algorithm. Comparatively, only Olivera and Viera [20] exhibit similar results. No feasible solutions are found on G3 instances.

5.2.2 Detailed results

The algorithm is run again five times over all the instances, but it now stops when a maximum number of iterations is performed or when the optimum value is found. It has been decided to terminate after 2000 crossovers are performed, since preliminary computational experiments shown that it is a good

Instance			Algorithm						
Name	m	T_H	TLG	BM	SP	OV	AAB	MAMTVRP-F	#fs
CMT1	3	192	✓	✓	✓	✓	✓	✓	5
CMT2	6	146	×	×	×	✓	×	✓	1
	7	131	✓	✓	×	✓	✓	✓	5
CMT3	4	217	✓	✓	×	✓	✓	✓	5
	5	173	×	✓	×	✓	✓	✓	5
	6	145	✓	✓	×	✓	✓	✓	5
	5	182	✓	✓	✓	✓	✓	✓	5
	6	151	✓	✓	✓	✓	✓	✓	5
CMT4	1	1080	✓	✓	✓	✓	✓	✓	5
	2	540	✓	✓	✓	✓	✓	✓	5
	3	360	✓	✓	×	✓	✓	✓	5
	4	270	✓	✓	×	✓	✓	✓	5
	5	216	✓	✓	×	✓	✓	✓	5
	6	180	✓	✓	×	✓	✓	✓	5
	8	135	×	×	×	✓	×	✓	2
	1	1131	✓	✓	✓	✓	✓	✓	5
	2	566	✓	✓	✓	✓	✓	✓	5
	3	377	✓	✓	✓	✓	✓	✓	5
	4	283	✓	✓	✓	✓	✓	✓	5
	5	226	✓	✓	✓	✓	✓	✓	5
	6	189	✓	✓	✓	✓	✓	✓	5
	7	162	✓	✓	×	✓	✓	✓	5
	8	141	✓	✓	×	✓	✓	✓	5
	CMT5	1	1356	✓	✓	✓	✓	✓	✓
2		678	✓	✓	✓	✓	✓	✓	5
3		452	✓	✓	×	✓	✓	✓	5
4		339	✓	✓	×	✓	✓	✓	5
5		271	✓	✓	×	✓	✓	✓	5
6		226	✓	✓	×	✓	✓	✓	5
7		194	✓	✓	×	✓	✓	✓	5
8		170	✓	✓	×	✓	✓	✓	5
9		151	✓	×	×	✓	×	✓	4
10		136	×	×	×	✓	×	✓	2
1		1421	✓	✓	✓	✓	✓	✓	5
2		710	✓	✓	✓	✓	✓	✓	5
3		474	✓	✓	✓	✓	✓	✓	5
4		355	✓	✓	✓	✓	✓	✓	5
5		284	✓	✓	✓	✓	✓	✓	5
6		237	✓	✓	✓	✓	✓	✓	5
7		203	✓	✓	✓	✓	✓	✓	5
8		178	✓	✓	×	✓	✓	✓	5
9		158	✓	✓	×	✓	✓	✓	5
10		142	✓	✓	×	✓	✓	✓	5
CMT11	4	274	×	×	×	✓	×	✓	1
CMT12	5	172	✓	✓	×	✓	✓	✓	1
F11	1	254	✓	✓	×	✓	×	✓	5
	2	127	×	×	×	✓	×	✓	5
	1	266	✓	✓	✓	✓	×	✓	5
	2	133	✓	✓	✓	✓	×	✓	5
	3	89	✓	✓	✓	✓	×	✓	5
F12	1	1221	✓	✓	✓	✓	×	✓	5
	2	611	✓	✓	✓	✓	×	✓	5
	3	407	✓	✓	✓	✓	×	✓	5
	1	1279	✓	✓	✓	✓	×	✓	5
	2	640	✓	✓	✓	✓	×	✓	5
	3	426	✓	✓	✓	✓	×	✓	5
# problems solved			50	50	29	56	40	56	

Table 5: Feasibility check on the 56 instances in G2

compromise between solution quality and computational efficiency. Complete and detailed results are reported in Tables 6–9. Results from the MA without CLS are reported in columns indicated with MAMTVRP while those from the MA with CLS are given in columns indicated with MAMTVRP+CLS.

Table 6 reports results obtained on the 42 instances of G1. Optimal values are indicated in bold. MAMTVRP and MAMTVRP+CLS find optimal solutions on all the five runs in 23 cases, but the former finds the optimal value at least once in 32 cases while the latter in 36 cases. In general, MAMTVRP+CLS is more efficient in finding optimal solutions: they are obtained 136 times over 210 runs while MAMTVRP finds optimal solutions 129 times. Both procedures always find feasible solutions. Note that Salhi and Petch [27] do not find any optimal solution and is outperformed by both methods on all instances.

Results on instances of G2 are detailed in Table 7. Here, bold numbers are used to indicate best known values. MAMTVRP finds a feasible solution at least once over all instances and the procedure finds a feasible solution on all the five runs in 50 cases (out of 56) for a total of 260 feasible solutions out of 280 runs. Introducing the CLS improves the results. Feasible solutions are always found in 52 cases and at least 2 feasible solutions are found over the five runs for a total of 271 feasible solutions. Again, solutions found by the procedures are always better than those reported in Salhi and Petch [27].

Tables 8 and 9 report results on instances of G3. First of all, it can be noticed from Table 8 that MAMTVRP+CLS finds a new feasible solution for instance CMT4_ T_H^1 _7 (details can be found in Appendix A). On the other five instances (Table 9), direct comparison with other methods on values of infeasible solutions found is possible. MAMTVRP+CLS finds two new best known values for instances CMT2_ T_H^1 _7 and F11_ T_H^1 _3. For the latter, the new best known value is as well reached by MAMTVRP. On average, both methods outperform the others.

Averagely, MAMTVRP+CLS performs better than MAMTVRP as can be seen in the last columns of Tables 6, 7 and 9. This, together with the new feasible solution found for instance CMT4_ T_H^1 _7 by MAMTVRP+CLS, validates the usefulness and efficiency of the CLS.

5.2.3 Detailed comparison with Olivera and Viera [20]

A full comparison following the scheme proposed by Olivera and Viera [20] is proposed in Tables 10 and 11. Given a solution ξ , the value $GAP(\xi)$ is calculated as

$$GAP(\xi) = 100 \cdot \left(\frac{c(\xi)}{z^*} + 1 \right), \quad (6)$$

and results are reported accordingly. The number of runs ended with a feasible solution is reported for instances in G1 as in Olivera and Viera [20] since they did not have optimal values available. As it can be noticed, results obtained by MAMTVRP+CLS outperform those by Olivera and Viera [20]. On the other side, the algorithm proposed by Olivera and Viera [20] performs better than MAMTVRP-F. A probable reason is that MAMTVRP-F terminates the procedure as soon as a feasible solution is found, while Olivera and Viera [20] check for feasibility each 100 iterations of their procedure.

5.2.4 Computational times

A fair computational time comparison could not be performed as the machine relative speeds were not found for all the computers used by previous papers. Machines used in previous works are listed in Table 12. Original computational times, as well as those of our method, are reported in Table 13 (times are expressed in seconds). Furthermore, algorithm differences and inharmonious computational time reporting complicate comparison. In particular, Taillard et al. [30] perform their algorithm five times on each instance. If no feasible is found, it is run another time. Average time on all runs is reported. Brandão and Mercer [5] stop their procedure once a feasible solution is found and they report computational times over the runs where a feasible solution is found. Salhi and Petch [27] and Alonso et al. [1] stop their algorithm

Instance			Algorithm								
			MRT	SP	MAMTVRP			MAMTVRP+CLS			
Name	m	T_H	Optimal	Best	Best	Average	#opt	Best	Av	#opt	
CMT1	1	551	524.61	546.28	524.61	524.61	5	524.61	524.61	5	
	2	275	533.00	×	533.00	533.67	4	533.00	533.00	5	
	1	577	524.61	547.14	524.61	524.61	5	524.61	524.61	5	
	2	289	529.85	549.42	529.85	529.85	5	529.85	530.67	3	
	4	144	546.29	566.86	546.29	546.29	5	546.29	546.29	5	
CMT2	1	877	835.26	869.06	835.26	838.40	2	835.26	838.40	2	
	2	439	835.26	865.48	835.77	840.04	0	835.26	838.59	1	
	3	292	835.26	×	835.26	836.32	1	835.26	838.58	2	
	4	219	835.26	856.77	835.77	839.41	0	835.77	839.77	0	
	5	175	835.8	×	836.18	841.97	0	836.18	836.52	0	
	1	919	835.26	869.73	835.26	835.48	2	835.26	835.48	2	
	2	459	835.26	881.50	835.26	839.20	1	835.26	836.46	1	
	3	306	835.26	869.11	835.77	840.07	0	835.26	837.40	2	
	4	230	835.26	880.90	838.17	840.41	0	835.26	837.73	2	
	5	184	835.26	883.29	835.77	837.71	0	835.77	837.99	0	
	6	153	835.22	×	843.09	848.06	0	839.22	846.02	0	
	CMT3	1	867	826.14	845.33	826.14	827.96	1	826.14	827.96	1
2		434	826.14	850.65	826.14	827.96	0	826.14	827.75	2	
3		289	826.14	×	828.08	829.63	0	826.14	828.53	1	
1		909	826.14	845.33	829.45	829.53	0	829.45	829.53	0	
2		454	826.14	872.10	826.14	828.80	1	826.14	827.96	1	
3		303	826.14	869.48	826.14	828.94	1	827.39	829.09	0	
4		227	826.14	878.00	826.14	828.01	1	826.14	827.55	1	
CMT11	1	1094	1042.11	1088.26	1042.11	1042.11	5	1042.11	1042.11	5	
	2	547	1042.11	×	1042.11	1042.11	5	1042.11	1042.11	5	
	3	365	1042.11	×	1042.11	1042.11	5	1042.11	1042.11	5	
	5	219	1042.11	×	1042.11	1042.11	5	1042.11	1042.11	5	
	1	1146	1042.11	1088.26	1042.11	1042.11	5	1042.11	1042.11	5	
	2	573	1042.11	1110.10	1042.11	1042.11	5	1042.11	1042.11	5	
	3	382	1042.11	1088.56	1042.11	1042.11	5	1042.11	1042.11	5	
	4	287	1042.11	×	1042.11	1042.11	5	1042.11	1042.11	5	
	5	229	1042.11	1092.95	1042.11	1042.11	5	1042.11	1042.11	5	
CMT12	1	861	819.56	819.97	819.56	819.56	5	819.56	819.56	5	
	2	430	819.56	821.33	819.56	819.56	5	819.56	819.56	5	
	3	287	819.56	826.98	819.56	819.56	5	819.56	819.56	5	
	4	215	819.56	824.57	819.56	819.56	5	819.56	819.56	5	
	1	902	819.56	819.97	819.56	819.56	5	819.56	819.56	5	
	2	451	819.56	829.54	819.56	819.56	5	819.56	819.56	5	
	3	301	819.56	851.16	819.56	819.56	5	819.56	819.56	5	
	4	225	819.56	821.53	819.56	819.56	5	819.56	819.56	5	
	5	180	824.78	833.85	824.78	824.78	5	824.78	824.78	5	
	6	150	823.14	855.36	823.14	823.14	5	823.14	823.15	5	
	total optimal solutions found							129	136		
	average					838.85	840.01	838.64		839.62	
average GAP from optimal					0.05	0.19	0.03		0.15		

Table 6: Feasible solutions on the 42 instances in G1

Instance				Algorithm						
				SP	MAMTVRP			MAMTVRP+CLS		
Name	m	T_H	Best known	Best	Best	Av	#fs	Best	Av	#fs
CMT1	3	192	552.68	560.26	552.68	552.68	5	552.68	552.68	5
CMT2	6	146	858.58	×	858.58	858.58	1	859.16	859.42	3
	7	131	844.70	×	853.88	861.64	5	844.70	854.70	5
CMT3	4	217	829.45	×	829.45	829.65	5	829.45	829.45	5
	5	173	832.89	×	832.89	835.98	5	832.89	843.72	5
	6	145	836.22	×	836.22	837.06	5	836.22	836.22	5
	5	182	832.34	901.30	832.34	833.83	5	832.34	832.88	5
	6	151	834.35	861.76	834.35	834.83	5	834.35	834.35	5
CMT4	1	1080	1031.00	1064.06	1031.00	1034.22	5	1031.00	1034.22	5
	2	540	1031.07	1065.86	1032.65	1038.33	5	1031.07	1037.89	5
	3	360	1028.42	×	1029.56	1036.73	5	1028.42	1032.79	5
	4	270	1031.10	×	1036.25	1040.75	5	1031.10	1037.09	5
	5	216	1031.07	×	1032.69	1040.42	5	1031.07	1037.41	5
	6	180	1034.61	×	1043.42	1046.71	5	1034.61	1041.82	5
	8	135	1056.54	×	1056.93	1059.58	2	1056.54	1059.68	3
	1	1131	1031.07	1088.93	1031.07	1038.77	5	1031.07	1038.77	5
	2	566	1030.45	1070.50	1030.45	1037.29	5	1034.08	1040.39	5
	3	377	1031.59	1077.24	1031.63	1040.75	5	1031.59	1032.92	5
	4	283	1031.07	1119.05	1031.07	1034.69	5	1031.96	1036.33	5
	5	226	1030.86	1085.38	1033.05	1039.52	5	1030.86	1035.52	5
	6	189	1030.45	1112.03	1032.16	1038.62	5	1030.45	1037.10	5
	7	162	1036.08	×	1043.92	1047.87	5	1036.08	1043.60	5
8	141	1044.32	×	1044.71	1050.28	5	1044.32	1048.08	5	
CMT5	1	1356	1302.43	1347.34	1302.43	1308.27	5	1302.43	1308.27	5
	2	678	1302.15	1346.63	1302.15	1309.04	5	1306.26	1309.66	5
	3	452	1301.29	×	1301.41	1309.33	5	1301.29	1307.85	5
	4	339	1304.78	×	1308.93	1312.76	5	1304.78	1308.07	5
	5	271	1300.02	×	1307.78	1314.66	5	1300.02	1307.10	5
	6	226	1303.37	×	1303.37	1314.29	5	1308.40	1311.16	5
	7	194	1309.40	×	1315.41	1319.86	5	1309.40	1313.06	5
	8	170	1303.91	×	1310.48	1316.53	5	1303.91	1308.98	5
	9	151	1307.93	×	1329.86	1331.61	4	1307.93	1317.03	5
	10	136	1323.01	×	1326.54	1326.54	1	1323.01	1329.00	5
	1	1421	1299.86	1340.44	1299.86	1310.43	5	1299.86	1310.43	5
	2	710	1305.35	1399.65	1305.35	1310.98	5	1307.70	1314.05	5
	3	474	1301.03	1409.37	1301.03	1312.15	5	1308.76	1310.93	5
	4	355	1303.65	1397.60	1303.65	1311.19	5	1310.97	1312.40	5
	5	284	1300.62	1411.19	1308.04	1311.87	5	1300.62	1308.75	5
	6	237	1306.17	1377.07	1306.17	1308.49	5	1306.25	1311.40	5
	7	203	1301.54	1394.73	1311.35	1314.18	5	1301.54	1313.66	5
	8	178	1308.78	×	1311.93	1313.86	5	1308.78	1310.61	5
	9	158	1307.25	×	1312.28	1318.26	5	1307.25	1311.32	5
	10	142	1308.81	×	1312.04	1321.27	5	1308.81	1316.80	5
CMT11	4	274	1078.64	×	1080.12	1080.12	1	1078.64	1080.38	3
CMT12	5	172	845.56	×	849.89	849.89	1	845.564	847.727	2
F11	1	254	241.97	×	241.97	241.97	5	241.97	241.97	5
	2	127	250.85	×	250.85	250.85	5	250.85	250.85	5
	1	266	241.97	254.07	241.97	241.97	5	241.97	241.97	5
	2	133	241.97	254.07	241.97	241.97	5	241.97	241.97	5
	3	89	254.07	256.53	254.07	254.07	5	254.07	254.07	5
F12	1	1221	1162.96	1190.21	1162.96	1162.96	5	1162.96	1162.96	5
	2	611	1162.96	1194.24	1162.96	1162.96	5	1162.96	1162.96	5
	3	407	1162.96	1199.86	1162.96	1163.05	5	1162.96	1162.96	5
	1	1279	1162.96	1183.00	1162.96	1162.96	5	1162.96	1162.96	5
	2	640	1162.96	1199.64	1162.96	1162.96	5	1162.96	1162.96	5
	3	426	1162.96	1215.43	1162.96	1162.96	5	1162.96	1162.96	5
total feasible solutions found							260			271
average					1040.90	1044.70		1039.23	1043.11	

Table 7: Feasible solutions on the 56 instances in G2

Name	m	T _H	Best known	Algorithm										
				TLG	BM	SP	OV	AAB	MAMTVRP			MAMTVRP+CLS		
				Best	Best	Best	Best	Best	Best	Av	#fs	Best	Av	#fs
CMT1	3	184	-	×	×	×	×	×	-	-	-	-	-	-
CMT1	4	138	-	×	×	×	×	×	-	-	-	-	-	-
CMT2	7	125	-	×	×	×	×	×	-	-	-	-	-	-
CMT4	7	154	1068.59	×	×	×	×	×	-	-	-	1068.59	1068.59	1
CMT12	6	143	-	×	×	×	×	×	-	-	-	-	-	-
F11	3	85	-	×	×	×	×	✗	-	-	-	-	-	-

Table 8: Feasible solutions on the 6 instances in G3

Name	m	T _H	Best known	Algorithm									
				TLG	BM	SP	OV	AAB	MAMTVRP		MAMTVRP+CLS		
				best	best	best	best	best	Best	Av	Best	Av	
CMT1	3	184	569.54	579.48	575.73	586.32	573.4	569.54	569.54	569.54	569.54	569.54	
CMT1	4	138	564.07	565.27	564.07	632.54	564.07	564.1	564.07	564.07	564.07	564.07	
CMT2	7	125	866.58	878.29	896.57	1056.34	877.12	878.05	876.77	880.06	866.58	873.14	
CMT12	6	143	845.48	845.48	847.85	898.88	860.61	866.54	845.48	845.48	845.48	845.48	
F11	3	85	256.93	257.31	257.47	266.85	260.55	✗	256.93	256.93	256.93	256.93	
average				625.17	628.34	688.19	627.15	-	622.56	623.21	620.52	621.83	

Table 9: Non-feasible solutions on the 5 unsolved instances in G3

Instance			MAMTVRP-F					OV					MAMTVRP+CLS				
Name	m	T _H	Best	Av	Worst	StDv	#fs	Best	Av	Worst	StDv	#fs	Best	Av	Worst	StDv	#fs
CMT1	1	551	0.0	2.7	4.8	2.0	5	0.0	0.0	0.0	0.0	5	0.0	0.0	0.0	0.0	5
	2	275	1.6	3.2	4.2	1.2	5	1.6	1.7	2.2	0.3	5	1.6	1.6	1.6	0.0	5
	1	577	2.3	4.8	7.3	2.3	5	0.0	0.0	0.0	0.0	5	0.0	0.0	0.0	0.0	5
	2	289	4.6	6.9	9.3	1.7	5	1.0	1.1	1.6	0.3	5	1.0	1.2	1.6	0.3	5
	4	144	4.6	5.7	7.6	1.3	5	4.1	4.2	4.3	0.1	5	4.1	4.1	4.1	0.0	5
CMT2	1	877	2.8	3.4	3.8	0.5	5	0.0	0.2	0.3	0.1	5	0.0	0.4	0.7	0.4	5
	2	439	2.6	3.6	4.4	0.7	5	0.2	0.5	1.0	0.4	5	0.0	0.4	0.7	0.3	5
	3	292	3.3	3.7	4.4	0.5	5	0.2	0.5	1.3	0.5	5	0.0	0.4	0.8	0.4	5
	4	219	1.8	2.4	3.3	0.7	5	0.1	0.7	1.7	0.7	5	0.1	0.5	0.8	0.3	5
	5	175	2.6	2.9	3.6	0.4	5	1.2	1.6	2.1	0.4	5	0.1	0.2	0.2	0.0	5
	1	919	4.3	4.9	5.5	0.5	5	0.1	0.7	1.3	0.5	5	0.0	0.0	0.1	0.0	5
	2	459	3.7	4.2	4.6	0.4	5	0.1	0.5	1.0	0.4	5	0.0	0.1	0.5	0.2	5
	3	306	3.3	5.3	8.0	1.9	5	0.1	0.4	0.8	0.3	5	0.0	0.3	0.7	0.4	5
	4	230	4.2	6.0	9.6	2.2	5	0.1	0.2	0.6	0.2	5	0.0	0.3	0.5	0.3	5
	5	184	3.7	5.4	8.1	2.0	5	0.1	0.9	1.3	0.5	5	0.1	0.3	0.7	0.3	5
	6	153	3.3	4.9	6.0	1.0	5	2.5	3.4	4.8	0.9	5	0.5	1.3	2.4	0.7	5
CMT3	1	867	1.6	2.8	4.0	0.9	5	0.0	0.4	0.6	0.2	5	0.0	0.2	0.4	0.2	5
	2	434	1.2	3.0	4.4	1.5	5	0.4	0.5	1.0	0.2	5	0.0	0.2	0.4	0.2	5
	3	289	2.3	3.4	4.6	0.8	5	0.5	0.7	0.9	0.2	5	0.0	0.3	0.4	0.2	5
	1	909	2.1	3.3	4.7	1.1	5	0.4	0.5	0.7	0.2	5	0.4	0.4	0.4	0.0	5
	2	454	2.2	4.8	6.7	1.6	5	0.3	0.4	0.7	0.2	5	0.0	0.2	0.4	0.2	5
CMT11	3	303	2.5	5.3	8.9	2.6	5	0.2	0.4	0.4	0.1	5	0.2	0.4	0.4	0.1	5
	4	227	2.3	5.4	7.9	2.1	5	0.0	0.3	0.5	0.2	5	0.0	0.2	0.4	0.1	5
	1	1094	0.8	3.6	5.0	1.7	5	0.1	2.5	3.2	1.3	5	0.0	0.0	0.0	0.0	5
	2	547	0.8	2.8	4.4	1.5	5	3.0	3.1	3.2	0.1	5	0.0	0.0	0.0	0.0	5
	3	365	0.2	0.7	1.1	0.3	5	0.2	2.7	4.5	2.1	5	0.0	0.0	0.0	0.0	5
	5	219	0.4	1.5	2.3	0.8	5	0.1	0.8	1.3	0.4	5	0.0	0.0	0.0	0.0	5
	1	1146	3.8	7.1	10.0	3.0	5	0.2	2.5	3.1	1.3	5	0.0	0.0	0.0	0.0	5
	2	573	0.8	3.8	7.1	2.3	5	1.2	2.6	3.1	0.8	5	0.0	0.0	0.0	0.0	5
	3	382	1.0	4.1	5.0	1.7	5	0.1	0.8	3.3	1.4	5	0.0	0.0	0.0	0.0	5
CMT12	4	287	0.6	1.6	2.6	0.8	5	0.1	0.2	0.4	0.1	5	0.0	0.0	0.0	0.0	5
	5	229	1.3	2.5	4.4	1.3	5	0.1	0.3	0.6	0.2	5	0.0	0.0	0.0	0.0	5
	1	861	0.2	1.5	5.0	2.0	5	0.0	0.1	0.2	0.1	5	0.0	0.0	0.0	0.0	5
	2	430	0.3	1.8	4.0	1.7	5	0.0	0.1	0.2	0.1	5	0.0	0.0	0.0	0.0	5
	3	287	0.5	2.1	3.3	1.3	5	0.0	0.1	0.1	0.1	5	0.0	0.0	0.0	0.0	5
	4	215	0.3	0.6	1.1	0.3	5	0.0	0.0	0.0	0.0	5	0.0	0.0	0.0	0.0	5
	1	902	0.2	4.2	8.1	3.7	5	0.0	0.1	0.1	0.1	5	0.0	0.0	0.0	0.0	5
	2	451	0.8	4.0	6.0	2.5	5	0.0	0.1	0.1	0.1	5	0.0	0.0	0.0	0.0	5
	3	301	0.7	2.6	4.2	1.7	5	0.0	0.1	0.1	0.1	5	0.0	0.0	0.0	0.0	5
	4	225	0.9	2.0	5.2	1.8	5	0.0	0.1	0.1	0.1	5	0.0	0.0	0.0	0.0	5
total feasible solutions found			210					210					210				
	average		1.9	3.6	5.4	1.5		0.5	0.9	1.3	0.4		0.2	0.3	0.5	0.1	

Table 10: Comparison with OV on the 42 instances in G1

Instance			MAMTVRP-F					OV					MAMTVRP+CLS					
Name	m	T_H	Best	Av	Worst	StDv	#fs	Best	Av	Worst	StDv	#fs	Best	Av	Worst	StDv	#fs	
CMT1	3	192	6.7	7.5	8.8	1.0	5	5.4	5.5	6.0	0.3	5	5.4	5.4	5.4	0.0	5	
CMT2	6	146	2.9	2.9	2.9	-	1	2.6	2.6	2.6	-	1	2.9	2.9	2.9	0.0	3	
	7	131	3.7	5.4	7.2	1.5	5	3.6	4.2	4.7	0.4	5	1.1	2.3	2.9	0.7	5	
CMT3	4	217	1.4	3.4	4.6	1.3	5	0.4	0.8	1.2	0.3	5	0.4	0.4	0.4	0.0	5	
	5	173	2.2	2.9	3.4	0.6	5	2.6	3.2	4.1	0.6	5	0.8	2.1	2.5	0.7	5	
	6	145	1.2	1.8	2.3	0.4	5	1.2	2.0	3.0	0.7	5	1.2	1.2	1.2	0.0	5	
	5	182	2.6	4.1	5.2	1.1	5	0.8	1.0	1.9	0.5	5	0.7	0.8	0.8	0.0	5	
	6	151	2.6	5.2	7.8	1.9	5	1.1	1.7	1.9	0.4	5	1.0	1.0	1.0	0.0	5	
CMT4	1	1080	3.1	3.8	4.6	0.7	5	0.5	0.9	1.3	0.4	5	0.3	0.6	1.4	0.5	5	
	2	540	2.6	3.7	4.4	0.7	5	0.8	0.7	3.5	1.1	5	0.3	0.9	1.4	0.6	5	
	3	360	2.3	3.5	4.2	0.8	5	0.7	1.3	1.8	0.4	5	0.0	0.4	1.4	0.6	5	
	4	270	2.6	3.7	4.2	0.7	5	0.8	1.6	2.5	0.9	5	0.3	0.8	1.5	0.6	5	
	5	216	2.9	3.6	4.6	0.7	5	0.4	1.6	2.8	0.9	5	0.3	0.9	1.8	0.7	5	
	6	180	2.7	3.6	4.2	0.7	5	2.0	3.2	4.5	0.9	5	0.6	1.3	2.1	0.6	5	
	8	135	2.8	3.0	3.3	0.4	2	3.0	3.6	4.3	0.7	3	2.7	3.0	3.4	0.3	3	
	1	1131	4.5	6.5	8.3	1.8	5	1.0	1.3	1.6	0.2	5	0.3	1.0	1.6	0.7	5	
	2	566	3.5	5.2	5.8	1.0	5	0.9	1.5	2.1	0.5	5	0.6	1.2	1.5	0.4	5	
	3	377	3.9	5.6	6.7	1.1	5	0.5	1.1	1.6	0.4	5	0.3	0.4	0.5	0.1	5	
	4	283	4.1	5.4	6.5	0.8	5	1.0	1.7	2.5	0.5	5	0.3	0.8	1.2	0.4	5	
	5	226	4.7	5.8	6.8	0.9	5	0.8	1.5	2.3	0.6	5	0.2	0.7	1.8	0.6	5	
	6	189	5.6	6.6	7.7	0.9	5	0.4	1.4	2.0	0.7	5	0.2	0.8	1.7	0.7	5	
	7	162	4.7	6.8	8.5	1.5	5	2.6	3.3	3.9	0.6	5	0.7	1.5	2.4	0.7	5	
	8	141	3.1	5.9	8.6	2.2	5	3.5	4.4	5.6	0.8	5	1.5	1.9	2.2	0.3	5	
CMT5	1	1356	4.1	4.6	4.9	0.3	5	1.9	2.8	3.4	0.6	5	0.9	1.3	1.5	0.3	5	
	2	678	4.3	4.6	4.9	0.2	5	2.0	3.1	3.9	0.8	5	1.1	1.4	1.8	0.2	5	
	3	452	4.2	4.4	4.7	0.2	5	1.6	2.1	2.8	0.4	5	0.8	1.3	1.5	0.3	5	
	4	339	4.2	4.3	4.7	0.2	5	2.5	3.0	3.2	0.3	5	1.0	1.3	1.5	0.2	5	
	5	271	4.0	4.2	4.6	0.3	5	2.5	3.1	4.1	0.2	5	0.7	1.2	1.5	0.4	5	
	6	226	3.5	3.8	4.5	0.4	5	3.1	3.5	4.0	0.4	5	1.3	1.5	1.9	0.2	5	
	7	194	2.9	3.7	4.3	0.6	5	3.6	3.8	4.1	0.2	5	1.4	1.7	2.1	0.3	5	
	8	170	3.3	3.7	4.4	0.5	5	2.8	3.5	4.2	0.6	5	1.0	1.4	1.8	0.3	5	
	9	151	3.6	4.0	4.5	0.5	4	3.4	3.7	4.0	0.3	4	1.3	2.0	2.8	0.6	5	
	10	136	3.5	3.8	4.1	0.4	2	4.1	4.1	4.1	-	1	2.4	2.9	3.6	0.5	5	
	1	1421	6.5	7.6	8.5	0.8	5	2.1	2.7	3.5	0.7	5	0.7	1.5	2.0	0.6	5	
	2	710	5.3	6.1	7.1	0.8	5	1.8	2.4	2.9	0.5	5	1.3	1.8	2.0	0.3	5	
	3	474	4.6	6.7	8.6	1.4	5	1.6	2.3	2.6	0.4	5	1.3	1.5	1.6	0.1	5	
	4	355	4.9	5.9	8.2	1.4	5	1.3	2.4	3.6	1.0	5	1.5	1.6	1.9	0.2	5	
	5	284	7.0	7.5	8.1	0.5	5	2.0	2.7	3.3	0.5	5	0.7	1.3	1.9	0.5	5	
	6	237	4.4	6.9	8.1	1.4	5	1.5	2.7	4.1	1.0	5	1.1	1.5	2.0	0.3	5	
	7	203	5.8	6.8	7.9	0.9	5	2.3	3.2	3.7	0.6	5	0.8	1.7	2.2	0.5	5	
	8	178	5.7	7.1	8.2	1.3	5	1.8	2.0	2.3	0.2	5	1.3	1.5	1.6	0.1	5	
	9	158	5.2	6.4	7.8	0.9	5	3.0	3.4	4.2	0.5	5	1.2	1.5	1.9	0.3	5	
	10	142	4.5	5.9	6.7	0.9	5	4.1	4.6	5.0	0.4	5	1.3	2.0	2.3	0.4	5	
CMT11	4	274	4.4	4.4	4.4	-	1	4.2	4.3	4.4	0.1	2	3.5	3.7	4.0	0.3	3	
CMT12	5	172	3.7	3.7	3.7	-	1	3.1	3.3	3.1	0.1	2	3.2	3.4	3.7	0.4	5	
F11	1	254	1.7	3.1	4.2	1.0	5	0.0	0.3	0.7	0.3	5	0.0	0.0	0.0	0.0	5	
	2	127	3.9	4.2	4.5	0.2	5	4.2	4.3	4.3	0.1	4	3.7	3.7	3.7	0.0	5	
	1	266	1.9	4.8	6.8	1.8	5	0.0	0.4	0.7	0.3	5	0.0	0.0	0.0	0.0	5	
	2	133	5.2	6.9	9.3	1.7	5	0.0	0.8	1.8	0.8	5	0.0	0.0	0.0	0.0	5	
	3	89	5.0	5.5	7.2	1.0	5	5.4	7.0	8.2	1.3	5	5.0	5.0	5.0	0.0	5	
F12	1	1221	1.3	3.8	4.9	1.4	5	0.5	0.6	0.9	0.2	5	0.0	0.0	0.0	0.0	5	
	2	611	2.7	3.9	4.6	0.9	5	0.7	0.9	1.1	0.2	5	0.0	0.0	0.0	0.0	5	
	3	407	1.1	2.3	3.9	1.2	5	0.3	0.5	0.7	0.2	5	0.0	0.0	0.0	0.0	5	
	1	1279	1.0	4.5	7.8	2.7	5	0.8	1.0	1.3	0.2	5	0.0	0.0	0.0	0.0	5	
	2	640	1.0	4.7	8.9	3.0	5	0.7	0.9	1.0	0.1	5	0.0	0.0	0.0	0.0	5	
	3	426	0.4	5.6	8.8	3.2	5	0.4	0.8	1.0	0.3	5	0.0	0.0	0.0	0.0	5	
total feasible solutions found			256					257					269					
average			3.6	4.9	6.0	1.0		1.9	2.4	3.0	0.5		1.1	1.4	1.8	0.3		

Table 11: Comparison with OV on the 56 instances in G2

Paper	Machine	RAM
TLG	100 Mhz Silicon Graphics Indingo	-
BM	HP Vectra XU Pentium Pro 200 Mhz	-
SP	Ultra Enterprise 450 dual processor 300 Mhz	-
OV	1.8 Ghz AMD Athlon XP 2200+	480 Mb
AAB	DELL Dimensio 8200 1.6 Ghz	256 Mb

Table 12: Machines used in previous papers

when a maximum number of iteration is reached, but while the former report average computational time over five runs the latter run the algorithm just once. Olivera and Viera [20] check for feasibility each 100 iterations and terminate the computation in case of success. They report computational times only for the best run. Finally, our goal is to find high quality solution and not to just satisfy feasibility as it was done in previous works. Keeping that in mind, it can be noticed from Table 13 that MAMTVRP-F is able to find feasible solutions very quickly (almost instantaneously for instances of families CMT3 and F12). We can also notice that the use of the CLS increases the time spent by the procedure. The time increase is, however, rewarded by more efficiency in finding optimal and feasible solutions as already outlined in Section 5.2.2.

Instance		Algorithm							
Name	#	TLG	BM	SP	OV	AAB	MAMTVRP-F	MAMTVRP	MAMTVRP+CLS
CMT1	8	300	150	16	16	161	3	5	14
CMT2	14	420	300	30	29	221	4	24	81
CMT3	12	1440	600	70	27	459	1	50	119
CMT4	16	3060	1500	206	68	681	31	169	493
CMT5	20	3960	3750	484	125	870	37	354	1284
CMT11	10	2700	1500	1132	28	527	12	98	220
CMT12	12	1380	600	45	27	414	10	16	50
F11	6	1560	150	93	13	✗	5	21	40
F12	6	4500	4800	584	31	✗	0	87	160

Table 13: CPU times comparison. Times expressed in seconds

6 Conclusion and future work

In this paper we proposed a genetic algorithm for the Multi Trip Vehicle Routing Problem. It is the first evolutionary procedure that efficiently faces the benchmark of instances proposed in the literature.

We use an adaptation of the *Split* procedure proposed by Prins [22] to evaluate the chromosomes.

We introduce a new LS operator that performs pejorative moves along with re-assignment of trips to vehicles and is called Combined LS (CLS). The efficiency of the CLS is validated by the quality of the results obtained. This opens a new promising research direction related to the management of moves combined with re-packing procedures.

We report detailed results over all instances (and not only for unsolved instances) and we give precise values of the found solutions (differently than what is done in Olivera and Viera [20]).

The method finds a feasible solution over 99 instances, one more than all the previous works (that have failed in finding a feasible solution for instance CMT4_ T_H^1 _7). Solutions found are always better than those reported by Salhi and Petch [27] (the only paper which gives detailed results). *GAP* values are averagely better than those reported by Olivera and Viera [20].

The proposed algorithm could be extended to the MTVRP with time windows introducing slight modifications into the *AdSplit* procedure explained in Section 3.2, in moves M1–M10 and in the CLS. This will be the subject of future research.

Acknowledgement

This work is supported by the French National Research Agency (ANR - Agence Nationale de la Recherche) and is part of project MODUM - Mutualisation et Optimisation de la distribution Urbaine de Marchandises.

References

- [1] F. Alonso, M.J. Alvarez, and J.E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of Operational Research Society*, 59(7):963–976, 2008.
- [2] N. Azi, M. Gendreau, and J.-Y. Potvin. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178(3):755–766, 2007.
- [3] M. Battarra, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36(11):3041–3050, 2009.
- [4] J. Brandão and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing problem. *European Journal of Operational Research*, 100, 1997.
- [5] J. Brandão and A. Mercer. The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49, 1998.
- [6] M. Browne, J. Allen, and J. Leonardi. Evaluating the use of an urban consolidation centre and electric vehicles in central london. *IATSS Research*, 35(1):1–6, 2011.
- [7] N. Christofides, A. Mingozzi, and P. Toth. *The vehicle routing problem*, pages 315–338. Wiley, Chichester, 1979.
- [8] F. Cornillier, G. Laporte, F.F. Boctor, and J. Renaud. The petrol station replenishment problem with time windows. *Computers & Operations Research*, 36(3):919–935, 2009.
- [9] M.L. Fisher. Optimal solution for vehicle routing problem using minimum k -trees. *Operations Research*, 42(4):626–642, 1994.
- [10] B. Fleischmann. The vehicle routing problem with multiple use of vehicles. Technical report, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, 1990.
- [11] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.
- [12] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem - Last Advances and New Challenges*. Operations Research Computer Science Interfaces. Springer, 2008.

- [13] I. Gribkovskaia, B.O. Gullberg, K.J. Hovden, and S.W. Wallace. Optimization model for a live-stock collection problem. *International Journal of Physical Distribution & Logistics Management*, 36(2):136–152, 2006.
- [14] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
- [15] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naudi. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. Technical report, laboratoire d’Informatique de Robotique et de Microélectronique de Montpellier (LIRMM), 2011.
- [16] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [17] A. Mingozzi, R. Roberti, and P. Toth. An exact algorithm for the multi-trip vehicle routing problem. *INFORMS Journal on Computing*, 00(0):1–27, 2012.
- [18] P. Moscato and C. Cotta. A modern introduction to memetic algorithms. In *Handbook of Metaheuristics - Second Edition*, International series in operations research and management science, chapter 6, pages 141–183. Springer, 2010.
- [19] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.
- [20] A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47, 2007.
- [21] R.J. Petch and S. Salhi. A multi-phase constructive heuristic for the vehicle routing problem with multi trips. *Discrete Applied Mathematics*, 133(1–3):69–92, 2004.
- [22] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.
- [23] C Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6):916–928, 2009.
- [24] C. Prins, N. Labadi, and M. Reghioui. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47(2):507–535, 2009.
- [25] C.R. Reeves. Genetic algorithms. In *Handbook of Metaheuristics - Second Edition*, International series in operations research and management science, chapter 5, pages 109–140. Springer, 2010.
- [26] Y. Rochat and É. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.
- [27] S. Salhi and R.J. Petch. A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modeling and Algorithms*, 6(4):591–613, 2007.
- [28] S.K. Smit and A.E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 399–406, 2009.
- [29] É.D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

- [30] É.D. Taillard, G. Laporte, and M. Gendreau. Vehicle routing with multiple use of vehicles. *Journal of Operational Research Society*, 47(8):1065–1070, 1996.
- [31] E. Taniguchi and H. Shimamoto. Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C*, 12(3), 2004.
- [32] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. Siam, 2002.
- [33] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal of Computing*, 15(4):333–346, 2003.
- [34] T. Vidal, T.G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. *Computers & Operations Research*, 40(1):475–489, 2013.
- [35] T. Vidal, T.G. Crainic, M. Gendreau, and W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.
- [36] P.C. Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21(2):281–283, 1970.

A New feasible solution for CMT4_ T_H^1 _7

The procedure found a new feasible solution for problem CMT4_ T_H^1 _7 that is detailed in Table 14 where v , r , τ_r and l_r indicate the vehicle, the route, its travelling time and its load.

v	r	τ_r	l_r	
1	1	152.00	195	0, 18, 60, 84, 114, 8, 46, 124, 47, 36, 143, 49, 64, 11, 126, 63, 90, 70, 101, 69, 0
2	1	150.42	200	0, 51, 103, 71, 65, 136, 35, 135, 34, 78, 121, 29, 24, 134, 25, 55, 130, 54, 0
3	1	97.33	200	0, 40, 73, 75, 56, 23, 67, 39, 139, 4, 110, 149, 26, 0
	2	55.68	174	0, 53, 138, 12, 109, 80, 150, 68, 116, 76, 111, 27, 0
4	1	73.68	196	0, 50, 102, 33, 81, 9, 120, 129, 79, 3, 77, 28, 0
	2	80.10	198	0, 146, 52, 106, 7, 82, 48, 123, 19, 107, 62, 148, 88, 127, 0
5	1	56.16	187	0, 96, 104, 99, 93, 85, 61, 5, 118, 89, 0
	2	95.96	199	0, 132, 1, 122, 30, 20, 66, 128, 131, 32, 108, 10, 31, 0
6	1	89.36	200	0, 59, 98, 91, 16, 141, 86, 113, 17, 45, 125, 83, 0
	2	64.60	156	0, 105, 21, 72, 74, 133, 22, 41, 145, 115, 2, 58, 0
7	1	36.35	130	0, 112, 147, 6, 94, 95, 117, 13, 0
	2	116.94	200	0, 137, 87, 144, 57, 15, 43, 42, 142, 14, 38, 140, 44, 119, 100, 37, 92, 97, 0

Table 14: New feasible solution for CMT4_ T_H^1 _7

