Adaptive Memory Programming for the Vehicle Routing Problem with Multiple Trips

Alfredo Olivera *, Omar Viera

Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Herrera y Reissig 525, Montevideo, Uruguay

Abstract

The Vehicle Routing Problem with Multiple Trips is an extension of the classical Vehicle Routing Problem in which each vehicle may perform several routes in the same planning period. In this paper, an adaptive memory algorithm to solve this problem is proposed. The algorithm was run over a set of benchmark problem instances, consistently finding high-quality solutions.

Key words: Vehicle Routing, Multiple Trips, Adaptive Memory Procedure

1 Introduction

The problem of distributing goods from depots to final consumers plays an important role in the management of many distribution systems, and its adequate programming may produce significant savings. The *Vehicle Routing Problem* (VRP) and its many variations have been subject of research during the last four decades. Some well studied characteristics include the existence of demands, time windows and heterogeneous vehicles [1].

However, some aspects that arise in real applications have not received much attention in the Operations Research literature. In most of the studied models, for instance, each vehicle is allowed to perform at most one route and the number of vehicles is supposed to be unlimited. In many contexts, this assumptions are unrealistic. For example, when the vehicle fleet is small or

* Corresponding author.
Email addresses: aolivera@fing.edu.uy (Alfredo Olivera), viera@fing.edu.uy (Omar Viera).

when the planning day is large with respect to the route duration, some vehicles may perform several routes in the same day. The *Vehicle Routing Problem with Multiple Trips* (VRPMT) overcomes the mentioned limitations, besides considering the classic VRP constraints. In this paper we describe a heuristic to solve the VRPMT, which is based on the Adaptive Memory Procedure (AMP) proposed by Rochat and Taillard [2].

A definition of the VRPMT is given in Section 2, as well as a literature review. The proposed algorithm is described in Section 3. Computational behavior of the algorithm is reported and analyzed in Section 4, while conclusions and future work are considered in Section 5.

2 The VRP with Multiple Trips

Let $G = (V, E)$ be a graph where $V = \{0, 1, \ldots, n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of arcs. If $(i, j) \in E$, then it is possible to travel from i to j, incurring in a cost c_{ij} and a travel time t_{ij}. Node 0 represents a depot where a fleet $K = \{1, \ldots, m\}$ of identical vehicles is based. Each vehicle has a limited capacity Q. The nodes in $V \setminus \{0\}$ represent customers, each one having a demand q_i. Finally, there exists a time horizon, denoted by T, which establishes the duration of a working day. It is assumed that Q, q_i and T are nonnegative integers.

The VRPMT calls for the determination of a set of routes and an assignment of each route to one vehicle, which minimizes the total routing costs and satisfies the following conditions:

(1) each route starts and ends at the depot,
(2) each customer is visited by exactly one route,
(3) the demand of the customers in the same route does not exceed Q,
(4) the duration of routes assigned to the same vehicle does not exceed T.

The VRPMT is a generalization of the VRP: any VRP instance can be transformed to an equivalent VRPMT instance, setting $m = n$ and $T = \sum_{(i,j)\in E} t_{ij}$. As the VRP is an NP-Hard problem [3], the VRPMT is also NP-Hard. The computational complexity of the VRPMT justifies the use of heuristics to solve instances of realistic size in moderate computation times.

2.1 Literature review

Fleischmann [4] was first one to address the problem. He proposed a modification to the savings algorithm [5] and used a Bin Packing Problem (BPP) [6]

heuristic to assign the routes to the vehicles.

Taillard et al. [7] proposed a three phase algorithm to solve the problem. In the first phase, a set of routes satisfying capacity constraints is constructed. Next, those routes are combined into complete VRP solutions. Finally, the routes of each solution are assigned to the vehicles solving a BPP.

Brandão and Mercer [8] designed a tabu search algorithm to solve a real life application which included, among others, the VRPMT features. Later, they simplified the algorithm [9] to handle the VRPMT. Key features are shared by both proposals. Moves are defined by swapping two customers and by removing one customer from its route and inserting it into another one. Insertions are made using the GENI algorithm [10]. Infeasible solutions are visited during the search and penalized using a modified objective function.

Petch and Salhi [11] proposed a multi-phase heuristic which first constructs many feasible VRP solutions and, for each one, assigns routes to vehicles using a BPP heuristic. Feasible VRP solutions are built with repeated executions of Yellow's savings algorithm [12] and, independently, using a route population approach.

2.2 Notation

A route will be denoted as a sequence of nodes $r = (v_0, \ldots, v_{n_r+1})$, where $v_0 = v_{n_r+1} = 0$ and $(v_i, v_{i+1}) \in E \ \forall\, i \in [0, n_r]$. For each route r, n_r is the number of customers it visits, $q_r = \sum_{i=1}^{n_r} q_{v_i}$ is the demand covered by the route, $c_r = \sum_{i=0}^{n_r} c_{v_i,v_{i+1}}$ is the cost and $t_r = \sum_{i=0}^{n_r} t_{v_i,v_{i+1}}$ is the duration. A route is feasible if $q_r \leq Q$.

A solution is a sequence of route sets $s = (R_1, \ldots, R_m)$, where R_k contains the routes assigned to vehicle k. The set of all the routes in a solution is $R(s) = \bigcup_{k \in K} R_k$. The cost of a solution is

$$f(s) = \sum_{r \in R(s)} c_r. \tag{1}$$

The total duration of routes assigned to each vehicle cannot be greater than the time horizon (these will be called *overtime constraints*). A measure of such constraint violation, called *overtime*, is defined for each vehicle:

$$O_k(s) = \left(\sum_{r \in R_k} t_r - T \right)^+ \tag{2}$$

where $a^+ = \max(a, 0)$. Finally, the *total overtime* of a solution is given by

$$O(s) = \sum_{k \in K} O_k(s). \tag{3}$$

2.3 Integer Programming Formulation

Let R be a set containing all feasible routes. For each $r \in R$ define a parameter a_{ir} indicating whether route r visits customer i ($a_{ir} = 1$) or not ($a_{ir} = 0$). The VRPMT can be formulated as an 0-1 Linear Program, as follows:

$$\min \sum_{k \in K} \sum_{r \in R} c_r x_r^k \tag{4}$$

$$\text{s.t.} \sum_{k \in K} \sum_{r \in R} a_{ir} x_r^k = 1 \qquad \forall\, i \in V \setminus \{0\} \tag{5}$$

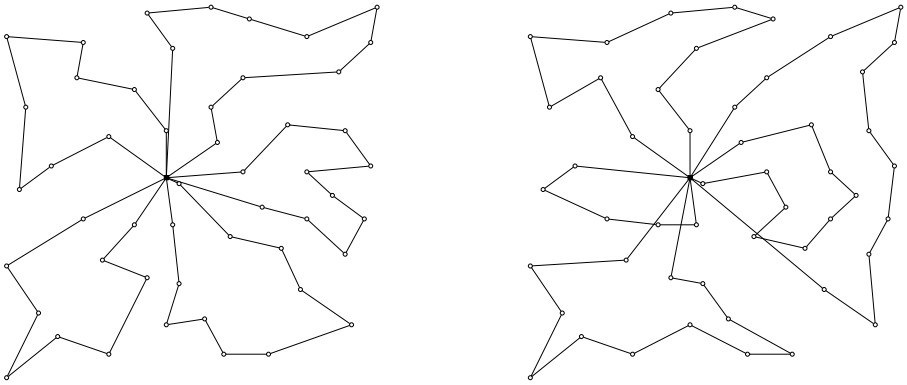
$$\sum_{r \in R} t_r x_r^k \leq T \qquad \forall\, k \in K \tag{6}$$

$$x_r^k \in \{0, 1\} \qquad \forall\, k \in K, \forall\, r \in R$$

This formulation resembles the Set Partitioning based VRP formulation due to Balinsky and Quandt [13]. If $x_r^k = 1$, then r is part of the solution and it is assigned to vehicle k (i.e. $r \in R_k$). If $x_r^k = 0 \,\forall\, k \in K$, r is not part of the solution. The objective function (4) establishes that the total routing cost should be minimized. The fact that each customer must belong to exactly one route is imposed in (5) and overtime constraints are established in (6).

Despite having an exponential number of variables, this formulation emphasizes the structure of VRPMT constraints. The bounded capacity of vehicles and the fact that routes must start and end at the depot are *local* to each route, and thus, can be encapsulated in the definition of R. Visiting each customer once and having no overtime are *global*, as they affect the whole solution. Violations to local constraints can be detected examining each route separately, but to ensure global constraints' satisfaction, the whole solution needs to be analyzed.

The presence of a fixed number of vehicles makes the overtime constraints hard to satisfy in the general case. On one hand, as there are only m vehicles available, the number of routes per vehicle tends to increase when m decreases. On the other hand, T imposes an upper bound on the total duration of routes assigned to each vehicle. For tight problems, feasible solutions can be hard to find and may not exist.



(a) Best known VRP solution (b) Feasible VRPMT solution

Fig. 1. Solutions for VRP and VRPMT formulations over CMT-1 problem.

2.4 Assigning routes to vehicles

The problem of assigning routes to the vehicles arises repeatedly when solving the VRPMT [4,7,11]. Given a set of routes $\{r_1, \ldots, r_p\}$, a feasible assignment of those routes to the vehicles can be obtained solving a BPP [6] with p items of weights $t_{r_1}, \ldots, t_{r_p}$ and m bins of capacity T.

However, a two phase approach that solves a VRP and assigns the obtained routes via a BPP may lead to infeasible solutions, as it is shown in the following example. Consider the VRP benchmark problem of 50 customers CMT-1, proposed by Christofides et al. [14]. The best known solution to that instance is shown in Figure 1(a), has a total cost of 524.61 and it is composed of 5 routes of durations $t_1 = 98.45$, $t_2 = 99.25$, $t_3 = 99.33$, $t_4 = 109.06$ and $t_5 = 118.52$. Consider a VRPMT instance over the same nodes, with the same vehicle capacity, $m = 4$ and $T = 144$. No feasible solution to this new instance can be constructed keeping the routes of the given VRP solution, because no packing of items of weights $t_1, \ldots, t_5$ in four bins of capacity 144 exists. However, a feasible solution to this instance exists, and an example is given in Figure 1(b). Its total cost is 546.29 and it has 5 routes of durations 57.82, 82.12, 122.80, 141.76 and 141.79, the first two being assigned to the same vehicle and the other three assigned each to a different one.

3 Adaptive Memory Algorithm

The Adaptive Memory Procedure (AMP) was first proposed by Rochat and Taillard [2] as an enhancement of Tabu Search (TS) to solve the VRP. It was motivated by the work of Glover regarding surrogate constraints [15]. An important principle behind AMP is that good solutions may be constructed by combining different *components* of other good solutions. A memory con-

taining components of visited solutions is kept. Periodically, a new solution is constructed using the data in the memory and improved by a local search procedure. The improved solution is then used to update the memory. A sketch of the AMP is given below:

(1) Initialize the memory M.
(2) While a stopping criteria is not met, do:
    (a) Construct a new solution s combining components of M.
    (b) Apply a local search procedure to s (let s^* be the improved solution).
    (c) Update M using components of s^*.

In the first iterations, the memory is expected to contain components of many different solutions. Thus, solutions constructed in Step 2a will differ substantially among iterations and diverse regions of the search space will be explored by the local search. As the whole procedure evolves, modifications made by the local search will be minor, making the components in the memory belong to a small set of good solutions. Thus, the search is intensified over promising regions.

AMP has been used to solve the VRP [2,16] and some of its variations, such as the Heterogeneous Fleet VRP [17] and the Multiple Depot VRP with Inter-Depot routes [18]. Other problems beyond vehicle routing have also been solved using AMP [19,20].

3.1 The algorithm

The algorithm is similar to the one proposed by Rochat and Taillard [2]. The memory M is a multi-set of routes. At every iteration, a new solution is constructed non-deterministically by selecting some of the routes in M. The solution is improved using a tabu search procedure, and the routes are added to M.

Infeasible solutions may be visited during the execution. In the TS, solutions may violate capacity and overtime constraints. In the rest of the AMP only overtime constraints may be violated.

The main cycle of the AMP is ran for AMP^{iter} iterations. In the rest of this section, each of the AMP components is explained: memory initialization, memory update, solution construction and tabu search procedure.

3.1.1 Memory initialization

To initialize the memory, I solutions are generated and inserted into M. Each solution is built running one iteration of the Sweep algorithm [21] starting from a randomly selected customer. The TS algorithm explained in Section 3.1.4 is used to improve each solution before inserting it in the memory.

3.1.2 Memory update

The memory is arranged in such a way that routes belonging to better solutions appear first. Solutions with positive overtime may be inserted. When a solution $s = (R_1, \ldots, R_m)$ is added to the memory, the following procedure is applied:

(1) Label each route $r \in R(s)$ with $(O(s), f(s))$.
(2) Add the routes to the memory: $M \leftarrow M \cup R(s)$.
(3) Sort M using a lexicographic ordering of route labels.
(4) If $|M| > M^{\text{size}}$ remove the last $|M| - M^{\text{size}}$ routes from M.

The first label of each route measures the infeasibility of the solution to which it belongs, while the second one is the cost of the solution. As a lexicographic criteria is used to sort, routes that belong to feasible solutions occupy the first positions (and within them, the ones that form part of least cost solutions go first).

If $|M|$ is too high, the probability of selecting good routes (see Section 3.1.3) may be diminished by the routes in the last positions. The parameter M^{size} specifies the maximum size of M. This bounding also contributes to space and time optimization of the AMP.

3.1.3 Solution construction

New solutions are built by probabilistically selecting routes from the memory. Higher probability is assigned to the routes in the first positions, because they belong to the best solutions found so far. The procedure is specified next:

(1) Set $M' \leftarrow M$.
(2) Probabilistically select $r \in M'$.
(3) Remove from M' all the routes having customers in common with r.
(4) If all customers have been routed, stop. If $M' \neq \emptyset$, go to Step 2, else go to Step 5.
(5) Solve a VRP with the unrouted customers and the depot using the Sweep algorithm (starting from a random customer) and add the routes to the solution.

The probability of selecting the i-th route of M' is $2\frac{|M'|+1-i}{|M'|(|M'|+1)}$.

3.1.4 Tabu Search procedure

A TS procedure is used to improve the constructed solution. Violations to capacity and overtime constraints are allowed, but penalized using the following objective function:

$$f_P(s) = f(s) + \alpha \sum_{r \in R(s)} (q_r - Q)^+ + \beta O(s). \tag{7}$$

The parameter α penalizes the total capacity constraints violation and β penalizes the total overtime. The values of α and β are updated at every iteration, as it is done in the Taburoute algorithm for the VRP [22], producing an oscillation between feasible and infeasible solutions.

Given a solution s, its neighborhood $N(s)$ is defined as the set of solutions that can be obtained applying one of the following operations to any pair of different routes $r_i = (\ldots, v_{i-1}, v_i, v_{i+1}, \ldots)$ and $r_j = (\ldots, w_{j-1}, w_j, w_{j+1}, \ldots)$ of $R(s)$:

Swap: set $r_i \leftarrow (\ldots, v_{i-1}, w_j, v_{i+1}, \ldots)$ and $r_j \leftarrow (\ldots, w_{j-1}, v_i, w_{j+1}, \ldots)$.
Insert: set $r_i \leftarrow (\ldots, v_{i-1}, v_{i+1}, \ldots)$ and $r_j \leftarrow (\ldots, w_{j-1}, v_i, w_j, w_{j+1}, \ldots)$.

In order to reduce the size of $N(s)$, and the running time of the search, a move is examined only if v_i is one of the p closest nodes to v_j and vice-versa. This reduced neighborhood will be called $N_p(s)$.

During the search, some moves are declared to be tabu and cannot be applied until its tabu status is revoked. After removing v from r, inserting it again in the same route is declared tabu for a random number of iterations. As usual, the tabu status of a move can be overridden if its application improves the best solution found so far (this is called *aspiration criteria*).

Let $N_p^*(s)$ be $N_p(s)$ restricted to moves that are not tabu or pass the aspiration criteria. Being s the incumbent solution, the whole $N_p^*(s)$ must be examined, and one solution minimizing f_P is set as the new incumbent solution.

After applying a move, a new assignment of routes to vehicles is computed by the heuristic given in Section 3.2. This is a major departure from previous approaches to the VRPMT, which make the assignment at the beginning [9] or at the final stage [7,11], but not at every iteration.

Whenever a new best solution is obtained, each of the routes that were modified by the last move is rearranged by means of the Unstringing and Stringing

(US) post-optimization procedure [10]. The parameter p of the US procedure was set to 5, which gives a good compromise between running time and solution quality [10]. The TS starts from a feasible VRP solution s and proceeds as follows:

(1) Assign the routes of s applying the heuristic given in Section 3.2.
(2) Initialize the tabu list. Set $\alpha \leftarrow \alpha^{\min}$, $\beta \leftarrow \beta^{\min}$ and $s^{\text{best}} \leftarrow s$.
(3) Repeat TS^{iter} times
    (a) Find $s' \in N_p^*(s)$ minimizing f_P.
    (b) If v was removed from r, inserting v in r is declared tabu for the next θ iterations, where θ is drawn uniformly in $[\theta^{\min}, \theta^{\max}]$.
    (c) Assign the routes of s' applying the heuristic given in Section 3.2.
    (d) If s' is better than s^{best}, apply the US procedure to the every route in s' and set $s^{\text{best}} \leftarrow s'$.
    (e) Update α and β.
    (f) Set $s \leftarrow s'$.
(4) Return s^{best}.

In Step 3d, s' is better than s^{best} if the routes of s' satisfy the capacity constraints and one of the following conditions hold:

- s' is "less infeasible" than s^{best}: $O(s') < O(s^{\text{best}})$
- both are feasible, but s' has a smaller cost: $O(s') = O(s^{\text{best}}) = 0$ and $f(s') < f(s^{\text{best}})$

The parameter update of Step 3e is done as follows. If s' satisfies the capacity constraints, then α is divided by 2, if not, it is multiplied by 2. The same is done with β, but with respect to overtime constraints. Lower and upper bounds $\alpha^{\min}$, $\beta^{\min}$, $\alpha^{\max}$ and $\beta^{\max}$ are imposed to α and β.

3.2 Assignment heuristic

During the TS, a VRPMT solution $s = (R_1, \ldots, R_m)$ must be constructed from a set of routes R'. As it has been explained in Section 2.4, this can be achieved solving a BPP. The Best Fit Decreasing (BFD) is a simple heuristic to solve the BPP [6]. The following modified BFD is used in this proposal:

(1) Initialize the assignment: $R_k \leftarrow \emptyset \; \forall \, k \in K$.
(2) Select the longest route that has not been assigned: $r^* \leftarrow \arg\max_{r \in R'} t_r$.
(3) Select the emptiest vehicle: $k^* \leftarrow \arg\min_{k \in K} \sum_{r \in R_k} t_r$.
(4) Assign r^* to k^*: $R_k \leftarrow R_k \cup \{r^*\}$.
(5) Disregard r^*: $R' \leftarrow R' \setminus \{r^*\}$.
(6) If $R' = \emptyset$, stop, else, go to Step 2.

The obtained solution is not guaranteed to satisfy the overtime constraints. When the resulting assignment is infeasible the following procedure is run, trying to obtain a feasible solution:

(1) Select the vehicle k^* which finishes later: $k^* = \arg\max_{k \in K} \sum_{r \in R_k} t_r$.
(2) Select $r^* \in R_{k^*}$ that has not been selected yet. If all routes in R_{k^*} have been considered, stop.
(3) If there exists $k' \in K$ and $r' \in R_{k'}$ such that:
$O_{k'} = 0$, $t_{r'} < t_{r^*}$ and $\sum_{r \in R_{k'}} t_r - t_{r'} + t_{r^*} \leq T$
go to Step 4. If not, go to Step 2.
(4) Set $R_{k^*} \leftarrow R_{k^*} \cup \{r'\} \setminus \{r^*\}$ and $R_{k'} \leftarrow R_{k'} \cup \{r^*\} \setminus \{r'\}$. If the solution is feasible, stop. If not, go to Step 1.

This algorithm tries to reduce the duration associated to the vehicle that arrives later, by exchanging a route with another vehicle. After routes are swapped in Step 4, vehicle k^* finishes earlier than before and k' finishes later (but within the time horizon). Again, feasibility is not guaranteed at this stage.

4 Computational experiments

4.1 Benchmark problems

The algorithm was tested over the 104 benchmark problems proposed by Taillard et al. [7]. These were constructed using the same graphs, demands and vehicle capacities than the problems 1-5 and 11-12 proposed by Christofides et al. [14] and problems 11 and 12 proposed by Fisher [23] for the VRP. For each VRP instance, several values of m are used. For each m, two time horizons $T_1 = [1.05\, z^*/m]$ and $T_2 = [1.10\, z^*/m]$ are proposed, where z^* is the best known solution value for the VRP instance and $[a]$ denotes a rounded to the nearest integer. All the problems are Euclidean and satisfy $t_{ij} = c_{ij}$.

The quality of feasible solutions can be assessed comparing its value with the corresponding VRP best known solution value z^*, using the GAP measure:

$$GAP(s) = 100 \left(\frac{f(s)}{z^*} - 1 \right). \tag{8}$$

Overtime constraints ensure that feasible solutions satisfy $f(s) \leq mT$, which for problems with time horizon T_1, implies that GAP will be upper-bounded by a value near 5%. Analogously, for problems with T_2, the GAP upper bound will be close to 10%.

Three articles provide computational experience regarding these benchmark

problems [7,8,11]. In all cases, feasible solutions to some problem instances could not be found. The GAP is not an adequate quality measure for these cases, and special measures must be defined. The total overtime (3) measures the time spent beyond the planning horizon. The longest tour ratio

$$LTR(s) = \frac{\max_{k \in R(s)} \sum_{r \in R_k} t_r}{T} \tag{9}$$

considers the duration of the vehicle which arrives latest in terms of the planning horizon. The penalized cost measures the routing cost that results from increasing c_{ij} by a factor of δ after the time horizon has elapsed:

$$PC(s) = f(s) + (\delta - 1)O(s). \tag{10}$$

The algorithm was coded in C++. All tests were done on a 1.8 GHz AMD Athlon XP 2200+ processor with 480 MBytes of RAM under Windows XP.

4.2 Parameter tuning

As in most meta-heuristics, there are a number of parameters which may modify the algorithm's behavior, namely:

- The number of AMP iterations: AMP^{iter}.
- The maximum number of routes in the memory: M^{size}.
- The number of initial solutions: I.
- The number of TS iterations: TS^{iter}.
- The neighborhood size: p.
- The interval in which tabu tenures are drawn: $[\theta^{\min}, \theta^{\max}]$.
- The bounds of α and β: $\alpha^{\min}$, $\beta^{\min}$, $\alpha^{\max}$ and $\beta^{\max}$.

In order to reduce the tuning effort, some parameters that are believed to be less influencing on the whole algorithm's performance were arbitrarily fixed. Those values are $\alpha^{\min} = \beta^{\min} = 1$ and $\alpha^{\max} = \beta^{\max} = 100000$ for the bounds, $\theta^{\min} = 5$ and $\theta^{\max} = 10$ for the tabu tenure interval and $I = 20$ for the number of initial solutions.

For each of the remaining parameters a set of candidate values was proposed, from which the final value was to be selected: $AMP^{\text{iter}} \in \{100, 200, 300\}$, $M^{\text{size}} \in \{150, 250\}$, $TS^{\text{iter}} \in \{250, 500, 750\}$ and $p \in \{15, 30\}$. To select the final values, the AMP procedure was ran over a subset of six benchmark problems, with every possible candidate value combination. The test problems selected for tuning were CMT-1, CMT-4, CMT-5, CMT-5, CMT-12 and F-11, using time horizon T_1 and m = 2, 8, 9, 7, 5 and 2 respectively.

Table 1
Running times for configurations that found feasible solutions for all tuning instances.

AMP^{iter}	TS^{iter}	M^{size}	p	Min.	Max.
300	500	250	15	34	354
300	750	250	15	48	533
300	750	150	30	109	880
300	750	250	30	108	888

For each run we recorded the running time and whether a feasible solution was found or not. Only the parameter configurations which led to a feasible solution for the six instances were further analyzed. For each, the minimum and maximum running times (in seconds) over the tuning problems are given in Table 1.

The selected parameter values were those which reported lowest computation times and obtained a feasible solution for every tuning instance: $AMP^{\text{iter}} = 300$, $TS^{\text{iter}} = 500$, $M^{\text{size}} = 250$ and $p = 15$. During the parameter tuning, it has been observed that for all problems, except CMT-5 (with $m = 9$), a feasible solution was found within the first 200 iterations. So, every 100 iterations the best solution found so far will be checked for feasibility, and the algorithm will keep running only if the solution is not feasible.

4.3 Results

All the reported results correspond to a unique run of the algorithm with the selected parameter configuration over the whole benchmark set. Detailed results are specified in Appendix A. Summarized results are given in Table 2. Each row refers to the instances obtained from the same base VRP problem. For each time horizon, the following results are reported:

- Feas./Total: the number of feasible solutions found and the number of problem instances.
- GAP: the average GAP (8) considering only the problems in which a feasible solution was found.
- Time: The average running time in seconds.
- The last row shows the total of Feas./Total column and the average GAP.

Feasible solutions to 95 out of 104 problem instances, including 3 which were unsolved by the previous proposals [7,9,11], have been found. A feasible solution was found for each of the 52 instances with T_2, as it is the case with the procedure proposed by Brandão and Mercer [9]. Taillard et al. [7] and Petch

Table 2
Results for one run over the benchmark problem set.

Problem	T_1			T_2		
	Feas./Total	GAP	Time	Feas./Total	GAP	Time
CMT-1	2/4	0.80	24	4/4	2.66	13
CMT-2	5/7	0.96	39	7/7	1.77	27
CMT-3	6/6	1.28	31	6/6	0.67	30
CMT-4	7/8	1.37	89	8/8	1.75	65
CMT-5	9/10	3.15	164	10/10	2.60	126
CMT-11	4/5	1.82	32	5/5	0.77	27
CMT-12	5/6	0.66	28	6/6	0.22	27
F-11	2/3	2.10	20	3/3	2.75	12
F-12	3/3	0.68	37	3/3	0.71	37
	43/52	1.60	–	52/52	1.59	–

and Salhi [11] report 48 and 46 feasible solutions, respectively. Instances with T_1 are more difficult, as they have a shorter time horizon. The proposed AMP found 43 feasible solutions, Brandão and Mercer [9] found 37, Taillard et al. [7] 38 and Petch and Salhi [11] 30.

The algorithm finds high quality solutions. Feasible solutions found are, on average, within 1.60% of the VRP best known solution, being 1.55% the GAP standard deviation. As the hardest to achieve for the benchmark problem set is feasibility and as every feasible solution is close to the VRP best known solution (see Section 4.1), other authors do not report the cost of the feasible solutions found. Thus, no comparison of this aspect can be made.

There are nine instances, all of which use T_1 as time horizon, that could not be solved by the AMP, and nor by the previous approaches [7,9,11].

In Table 3 infeasible solutions are compared in terms of LTR values. Each row corresponds to an instance for which the AMP procedure could not find a feasible solution. The average value for each algorithm and the difference between AMP values and each other, are given in the last two rows. The algorithm obtains the best results in all but two instances. The average LTR value over the nine infeasible instances is significantly smaller than the one obtained by other authors. These results suggest that the proposed algorithm is highly accurate in finding routes which fit the time horizon.

Table 4 compares the cost of the infeasible solutions with those obtained by Brandão and Mercer [9] and Taillard et al. [7] (Petch and Salhi [11] do not

Table 3
Longest route ratio (LRR) comparison for infeasible solutions.

	Problem	m	AMP	Brandão and Mercer [9]	Taillard et al. [7]	Petch and Salhi [11]
1	CMT-1	3	1.030	1.041	1.115	**1.026**
2	CMT-1	4	**1.027**	**1.027**	**1.027**	1.085
3	CMT-2	6	**1.003**	1.031	1.032	1.019
4	CMT-2	7	**1.009**	1.088	1.073	1.064
5	CMT-4	7	**1.003**	1.071	1.033	1.072
6	CMT-5	10	**1.007**	1.051	1.024	1.064
7	CMT-11	4	**1.001**	1.011	1.020	1.052
8	CMT-12	6	**1.024**	1.072	1.064	1.029
9	F-11	3	1.020	**1.011**	1.075	1.020
		Average	**1.014**	1.045	1.051	1.048
		Average difference (%)	–	−2.91	−3.52	−3.20

report costs for the infeasible solutions). For each instance, the cost (1), the total overtime (3) and the penalized cost (10) using $\delta = 2$ are given. The last row shows, for each measure, the average difference between the AMP and each other algorithm.

The values must be analyzed carefully. As it was pointed in Section 2.4, fitting the time horizon and minimizing the routing costs may be opposed objectives. In the general case, it is unlikely that both can be simultaneously attained. On one hand, if high overtime values are admitted, solutions with smaller cost (but highly infeasible) may be found. On the other hand, if the admitted overtime is small, some customers may have to be relocated in order to shorten some of the longest routes, but increasing the total cost.

The routing costs given by Taillard et al. [7] are, in all cases, the smallest. However, the total overtime values obtained by our algorithm are lower. This is not surprising, since Taillard et al. [7] assign routes to vehicles in the last phase of their algorithm, while our AMP modifies the assignment at every iteration. While Taillard et al. [7] obtain low cost routes that are hard to pack in working days, our AMP produces solutions with slightly higher costs (3.51%) but which are dramatically better in terms of feasibility (73.21%). Overall, in terms of PC values, solutions obtained by the AMP algorithm are slightly outperformed by Taillard et al. [7] and Brandão and Mercer [9] when $\delta = 2$. However, the good compromise between infeasibility and cost obtained by the AMP is captured when using higher values of δ.

Table 4. Total overtime (O) and penalized cost (PC) comparison for infeasible solutions.

	AMP			Brandão and Mercer [9]			Taillard et al. [7]		
	Cost	O	PC	Cost	O	PC	Cost	O	PC
1	559.20	**8.00**	**575.20**	556.34	9.70	575.73	**533.00**	23.24	579.48
2	547.10	**8.49**	**564.07**	547.10	**8.49**	**564.07**	**546.29**	9.49	565.27
3	872.45	**0.57**	873.58	858.04	4.49	867.02	**841.60**	7.80	**857.19**
4	873.40	**1.86**	**877.11**	870.07	13.25	896.57	**843.60**	17.35	878.29
5	1073.79	**0.39**	1074.57	1063.95	19.34	1102.63	**1042.39**	15.89	**1074.16**
6	1356.59	**2.53**	1361.65	1336.92	11.42	1359.75	**1316.00**	7.55	**1331.09**
7	1082.61	**0.32**	1083.25	1069.24	3.06	1075.35	**1042.11**	6.48	**1055.07**
8	855.14	3.81	862.76	**819.56**	**3.19**	**825.94**	**819.56**	12.96	845.48
9	256.85	1.85	260.54	254.40	**1.54**	257.47	**244.60**	6.36	**257.31**
Average difference (%)				1.28	−46.29	0.27	3.51	−73.21	1.01

5 Conclusion

An algorithm based on the Adaptive Memory Programming principle has been proposed to solve the Vehicle Routing Problem with Multiple Trips, an important extension of the classic Vehicle Routing Problem.

The algorithm was ran over a set of benchmark problems and the solutions obtained were compared with those reported for three previously proposed algorithms [7,9,11]. Our algorithm obtains more feasible solutions than the previous approaches. Further analysis shows that a good compromise between routing cost and overtime violation is achieved for highly constrained instances.

The proposed algorithm may be extended to handle a heterogeneous fleet of vehicles, modifying the way that routes are assigned to vehicles. New benchmark problems can be constructed in the same way as the ones used in the article. Lower bounds should be derived and experimental analysis should be carried out. Handling time windows is a harder problem since the assignment of routes to vehicles cannot be solved via a Bin Packing Problem.

A Detailed computational results

Tables A.1, A.2 and A.3 present detailed results for each one of the 104 benchmark problem instances. The meaning of each column is:

- Problem: the base VRP problem.
- m and T: the number of vehicles and time horizon. For each base problem, instances with T_1 are listed first. Bold entries indicate that a new feasible solution was found by the AMP.
- Feas.: indicates whether a feasible solution was found (✓) or not (×).
- Cost: the routing cost of the solution.
- GAP: the measure defined in (8). It is reported for feasible solutions only.
- Iter.: the AMP iteration in which the reported solution was found.
- Time: the running time in seconds.
- PS, BM, TLG: indicates whether a feasible solution was found or not by Petch and Salhi [11], Brandão and Mercer [9] and Taillard et al. [7], respectively.

B New feasible solutions

Feasible solutions to problems for which all previous approaches were infeasible are presented next.

New feasible solution to CMT-3 with $m = 6$ and $T_1 = 145$.

Veh.	Duration	Node sequence
1	139.7459	$(0, 70, 30, 20, 66, 65, 71, 35, 34, 78, 33, 81, 9, 51, 1, 69, 0)$
2	139.2430	$(0, 89, 60, 83, 45, 8, 46, 36, 49, 64, 63, 90, 32, 10, 31, 27, 0)$
3	139.0635	$(0, 50, 3, 79, 29, 24, 54, 55, 25, 39, 67, 23, 56, 4, 0)$
4	137.0156	$(0, 2, 57, 15, 43, 42, 14, 44, 38, 86, 16, 61, 17, 84, 5, 0)$
5	93.2598	$(0, 18, 82, 48, 47, 19, 11, 62, 88, 7, 52, 0)$
5	51.4576	$(0, 28, 76, 77, 68, 80, 12, 26, 0)$
6	81.8540	$(0, 53, 58, 40, 21, 73, 72, 74, 75, 22, 41, 87, 97, 13, 0)$
6	58.2624	$(0, 6, 96, 99, 59, 93, 85, 91, 100, 37, 98, 92, 95, 94, 0)$

New feasible solution to CMT-4 with $m = 8$ and $T_1 = 135$.

Veh.	Duration	Sequence
1	134.5364	$(0, 110, 4, 56, 75, 23, 67, 39, 139, 25, 55, 24, 29, 121, 80, 109, 0)$
2	134.3165	$(0, 1, 122, 51, 120, 9, 103, 66, 71, 65, 136, 35, 135, 34, 78, 0)$
3	130.0318	$(0, 60, 84, 17, 113, 86, 140, 38, 44, 119, 14, 142, 42, 43, 15, \ldots$ $\ldots, 57, 144, 0)$
4	128.6711	$(0, 146, 7, 19, 107, 11, 64, 49, 143, 36, 47, 124, 46, 8, 114, 18, 0)$
5	89.9579	$(0, 132, 69, 101, 70, 30, 20, 128, 131, 32, 90, 63, 126, \ldots)$ $\ldots, 108, 10, 31, 0)$
5	44.9832	$(0, 112, 94, 95, 117, 97, 87, 137, 13, 0)$
6	62.9857	$(0, 28, 76, 116, 77, 3, 79, 129, 81, 33, 102, 50, 111, 0)$
6	70.5012	$(0, 6, 59, 93, 85, 61, 16, 141, 91, 100, 37, 98, 92, 0)$
7	64.2336	$(0, 105, 26, 149, 130, 54, 134, 150, 68, 12, 138, 0)$
7	70.2185	$(0, 27, 127, 88, 148, 62, 123, 48, 82, 106, 52, 0)$
8	66.4566	$(0, 58, 2, 115, 145, 41, 22, 133, 74, 72, 73, 21, 40, 53, 0)$
8	68.5059	$(0, 89, 118, 83, 45, 125, 5, 104, 99, 96, 147, 0)$

New feasible solution to F-11 with $m = 2$ and $T_1 = 127$.

Veh.	Duration	Sequence
1	97.0179	$(0, 9, 7, 4, 8, 3, 5, 10, 6, 71, 12, 16, 17, 13, 1, 15, 2, 19, 62, \ldots$ $\ldots, 64, 65, 63, 59, 57, 41, 55, 54, 0)$
1	28.3711	$(0, 35, 18, 11, 14, 31, 0)$
2	68.4026	$(0, 20, 29, 23, 26, 24, 25, 49, 51, 70, 50, 47, 48, 52, 45, 53, \ldots$ $\ldots, 46, 44, 43, 42, 27, 28, 22, 21, 30, 0)$
2	58.3373	$(0, 36, 33, 60, 61, 58, 66, 67, 69, 37, 38, 40, 68, 39, 56, 34, 32, 0)$

References

[1] P. Toth, D. Vigo, The vehicle routing problem, SIAM Monographs on Discrete Mathematics and Applications, 2002.

[2] Y. Rochat, E. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.

[3] J. Lenstra, A. Rinnooy Kan, Complexity of vehicle routing and scheduling problems, Networks 11 (1981) 221–227.

[4] B. Fleischmann, The vehicle routing problem with multiple use of vehicles, Working Paper. Fachbereich Wirtschaftswissenschaften, Universität Hamburg (1990).

[5] G. Clarke, W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (1964) 568–581.

[6] S. Martello, P. Toth, Knapsack Problems, Wiley, Chichester, 1990.

[7] E. Taillard, G. Laporte, M. Gendreau, The vehicle routing problem with multiple use of vehicles, Journal of the Operational Research Society 47 (1996) 1065–1070.

[8] J. Brandão, A. Mercer, A tabu search algorithm for the multi-trip vehicle routing and scheduling problem, European Journal of Operational Research 100 (1997) 180–191.

[9] J. Brandão, A. Mercer, The multi-trip vehicle routing problem, Journal of the Operational Research Society 49 (1998) 799–805.

[10] M. Gendreau, A. Hertz, G. Laporte, New insertion and post optimization procedures for the travelling salesman problem, Operations Research 40 (1992) 1086–1094.

[11] R. Petch, S. Salhi, A multi-phase constructive heuristic for the vehicle routing problem with multiple trips, Discrete Applied Mathematics 133 (2004) 69–92.

[12] P. Yellow, A computational modification to the savings method of vehicle scheduling, Operational Research Quarterly 21 (1970) 281–283.

[13] M. Balinski, R. Quandt, On an integer program for a delivery problem, Operations Research 12 (1964) 300–304.

[14] N. Christofides, A. Mingozzi, P. Toth, Combinatorial Optimization, Wiley, Chichester, 1979, Ch. The Vehicle Routing Problem, pp. 315–338.

[15] F. Glover, Heuristics for integer programming using surrogate constraints, Decision Sciences 8 (1977) 156–166.

[16] C. Tarantilis, C. Kiranoudis, BoneRoute: An adaptive memory-based method for effective fleet management, Annals of Operations Research 114 (2002) 227–241.

[17] M. Gendreau, G. Laporte, C. Musaraganyi, E. Taillard, A tabu search heuristic for the heterogeneous fleet vehicle routing problem, Computers and Operations Research 26 (1999) 1153–1173.

[18] B. Crevier, J.-F. Cordeau, E. Taillard, The multi-depot vehicle routing problem with inter-depot routes, Tech. Rep. CRT-2004-10, Centre de recherche sur les transports, Montreal (2004).

[19] B. Bozkaya, E. Erkut, G. Laporte, A tabu search heuristic and adaptive memory procedure for political districting, European Journal of Operational Research 144 (2003) 12–26.

[20] L. Hvattum, A. Løkketangen, F. Glover, Adaptive memory search for boolean optimization problems, Discrete Applied Mathematics 142 (2004) 99–109.

[21] B. Gillett, L. Miller, A heuristic algorithm for the vehicle-dispatch problem, Operations Research 22 (1974) 340–349.

[22] M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.

[23] M. Fisher, Optimal solution of vehicle routing problems using minimum k-trees, Operations Research 42 (1994) 626–642.

Table A.1
Detailed computational results for problems CMT-1, CMT-2 and CMT-3.

Problem	m	T	Feas.	Cost	GAP	Iter.	Time	PS	BM	TLG
CMT-1	1	551	✓	524.61	0.0	0	12	✓	✓	✓
$n = 50$	2	275	✓	533.00	1.6	56	13	×	✓	✓
$z^* = 524.61$	3	184	×	559.20	–	6	36	×	×	×
	4	138	×	547.10	–	78	34	×	×	×
	1	577	✓	524.61	0.0	4	12	✓	✓	✓
	2	289	✓	529.85	1.0	0	13	✓	✓	✓
	3	192	✓	552.68	5.1	0	14	✓	✓	×
	4	144	✓	547.10	4.1	112	13	×	✓	✓
CMT-2	1	877	✓	835.67	< 0.1	98	25	✓	✓	✓
$n = 75$	2	439	✓	843.13	0.9	94	26	✓	✓	✓
$z^* = 835.26$	3	292	✓	846.37	1.3	76	26	✓	✓	✓
	4	219	✓	838.71	0.4	89	27	✓	✓	✓
	5	175	✓	852.66	2.0	93	26	✓	✓	✓
	6	146	×	872.45	–	261	71	×	×	×
	7	125	×	873.40	–	264	74	×	×	×
	1	919	✓	844.26	1.1	55	26	✓	✓	✓
	2	459	✓	841.23	0.7	71	26	✓	✓	✓
	3	306	✓	836.77	0.2	78	27	✓	✓	✓
	4	230	✓	836.18	0.1	100	27	✓	✓	✓
	5	184	✓	844.28	1.1	100	28	✓	✓	✓
	6	153	✓	875.03	4.5	41	28	✓	✓	✓
	7	131	✓	872.64	4.3	99	28	×	✓	×
CMT-3	1	867	✓	830.77	0.6	40	29	✓	✓	✓
$n = 100$	2	434	✓	834.15	1.0	81	30	✓	✓	✓
$z^* = 826.14$	3	289	✓	831.16	0.6	77	31	✓	✓	✓
	4	217	✓	832.74	0.8	81	30	✓	✓	✓
	5	173	✓	851.47	3.0	87	32	×	✓	×
	6	**145**	✓	839.90	1.6	100	31	×	×	×
	1	909	✓	829.69	0.4	34	28	✓	✓	✓
	2	454	✓	829.54	0.4	75	30	✓	✓	✓
	3	303	✓	829.45	0.4	87	30	✓	✓	✓
	4	227	✓	826.14	0.0	95	30	✓	✓	✓
	5	182	✓	833.15	0.8	88	30	✓	✓	✓
	6	151	✓	842.21	1.9	100	33	✓	✓	✓

Table A.2
Detailed computational results for problems CMT-4, CMT-5 and CMT-11.

Problem	m	T	Feas.	Cost	GAP	Iter.	Time	PS	BM	TLG
CMT-4	1	1080	✓	1033.21	0.5	98	60	✓	✓	✓
$n = 150$	2	540	✓	1036.70	0.8	93	61	✓	✓	✓
$z^* = 1028.42$	3	360	✓	1035.48	0.7	63	63	✓	✓	✓
	4	270	✓	1036.35	0.8	72	63	×	✓	✓
	5	216	✓	1033.02	0.4	91	63	✓	✓	✓
	6	180	✓	1058.04	2.8	48	65	×	✓	✓
	7	154	×	1073.79	–	289	170	×	×	×
	8	**135**	✓	1064.97	3.4	214	164	×	×	×
	1	1131	✓	1041.77	1.3	87	61	✓	✓	✓
	2	566	✓	1047.02	1.8	63	63	✓	✓	✓
	3	377	✓	1038.98	1.0	83	62	✓	✓	✓
	4	283	✓	1038.88	1.0	100	62	✓	✓	✓
	5	226	✓	1044.09	1.5	97	66	✓	✓	✓
	6	189	✓	1033.02	0.4	61	65	✓	✓	✓
	7	162	✓	1062.89	3.2	64	68	×	✓	✓
	8	141	✓	1064.56	3.4	35	69	✓	✓	×
CMT-5	1	1356	✓	1323.13	2.4	97	120	✓	✓	✓
$n = 199$	2	678	✓	1341.41	3.7	90	122	✓	✓	✓
$z^* = 1291.44$	3	452	✓	1317.58	2.0	74	121	✓	✓	✓
	4	339	✓	1330.63	2.9	95	123	✓	✓	✓
	5	271	✓	1329.17	2.8	87	126	×	✓	✓
	6	226	✓	1337.05	3.4	92	123	✓	✓	✓
	7	194	✓	1340.91	3.7	73	127	×	✓	✓
	8	170	✓	1327.09	2.7	74	124	×	✓	✓
	9	151	✓	1342.50	3.8	273	330	×	×	✓
	10	136	×	1356.59	–	269	327	×	×	×
	1	1421	✓	1318.46	2.0	100	120	✓	✓	✓
	2	710	✓	1314.09	1.7	100	123	✓	✓	✓
	3	474	✓	1311.89	1.6	86	124	✓	✓	✓
	4	355	✓	1338.52	3.5	100	126	✓	✓	✓
	5	284	✓	1322.64	2.4	98	126	✓	✓	✓
	6	237	✓	1311.10	1.5	100	127	✓	✓	✓
	7	203	✓	1337.81	3.5	71	128	✓	✓	✓
	8	178	✓	1316.89	1.9	83	125	✓	✓	✓
	9	158	✓	1331.17	3.0	99	128	✓	✓	✓
	10	142	✓	1347.99	4.2	79	130	×	✓	✓
CMT-11	1	1094	✓	1073.34	2.9	57	26	✓	✓	✓
$n = 120$	2	547	✓	1073.07	2.9	90	28	×	✓	✓
$z^* = 1042.11$	3	365	✓	1047.97	0.6	95	28	×	✓	✓
	4	274	×	1082.61	–	300	48	×	×	×
	5	219	✓	1049.81	0.7	83	29	×	✓	✓
	1	1146	✓	1044.35	0.2	94	26	✓	✓	✓
	2	573	✓	1072.21	2.8	90	26	✓	✓	✓
	3	382	✓	1043.17	0.1	90	27	✓	✓	✓
	4	287	✓	1045.07	0.3	99	28	×	✓	✓
	5	229	✓	1045.85	0.4	93	30	✓	✓	✓

Table A.3
Detailed computational results for problems CMT-12, F-11 and F-12.

Problem	m	T	Feas.	Cost	GAP	Iter.	Time	PS	BM	TLG
CMT-12	1	861	✓	820.96	0.2	31	26	✓	✓	✓
$n = 100$	2	430	✓	819.56	0.0	27	27	✓	✓	✓
$z^* = 819.56$	3	287	✓	819.60	< 0.1	22	27	✓	✓	✓
	4	215	✓	819.56	0.0	91	28	✓	×	✓
	5	172	✓	845.37	3.1	91	27	✓	×	×
	6	143	×	855.14	–	95	30	×	×	×
	1	902	✓	819.56	0.0	37	26	✓	✓	✓
	2	451	✓	819.56	0.0	44	26	✓	✓	✓
	3	301	✓	819.56	0.0	21	27	✓	✓	✓
	4	225	✓	819.56	0.0	20	28	✓	✓	✓
	5	180	✓	824.78	0.6	86	28	✓	✓	✓
	6	150	✓	825.36	0.7	28	29	✓	✓	✓
F-11	1	254	✓	241.97	0.0	44	10	✓	✓	✓
$n = 71$	**2**	**127**	✓	252.13	4.0	184	21	×	×	×
$z^* = 241.97$	3	85	×	256.85	–	212	28	×	×	×
	1	266	✓	243.25	0.5	34	10	✓	✓	✓
	2	133	✓	241.97	0.0	58	12	✓	✓	✓
	3	89	✓	260.63	7.2	72	13	✓	✓	×
F-12	1	1221	✓	1171.16	0.7	41	35	✓	✓	✓
$n = 134$	2	611	✓	1175.30	1.1	55	36	✓	✓	✓
$z^* = 1162.96$	3	407	✓	1166.18	0.3	85	39	✓	✓	✓
	1	1279	✓	1173.07	0.9	75	37	✓	✓	✓
	2	640	✓	1173.18	0.9	77	35	✓	✓	✓
	3	426	✓	1167.43	0.4	97	38	✓	✓	✓