The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates

Diego Cattaruzza, Nabil Absi, Dominique Feillet

The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates

Diego CATTARUZZA, Nabil ABSI, Dominique FEILLET
Ecole des Mines de Saint-Etienne
Department of Manufacturing Sciences and Logistics,
CMP Georges Charpak, F-13541 Gardanne, France
{cattaruzza, absi, feillet}@emse.fr

January 2014

Working Paper EMSE CMP–SFL 2014/1

The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates is a variant of the Multi Trip Vehicle Routing Problem where a time windows is associated with each customer and a *release date* is associated with each merchandise to be delivered at a certain client. The release date represents the moment the merchandise becomes available at the depot for final delivery.

The problem is relevant in city logistics context, where delivery systems based on city distribution centers (CDC) are studied. Trucks arrive at the CDC during the whole working day to deliver goods that are transferred to eco-friendly vehicles in charge of accomplish final deliveries to customers.

We propose a population-based algorithm for the problem based on giant tour representation of the chromosomes as well as a *split* procedure to obtain solutions from individuals.

The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates

Diego Cattaruzza[1], Nabil Absi[1], and Dominique Feillet[1]

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

Abstract

The Multi Trip Vehicle Routing Problem with Time Windows and Release Dates is a variant of the Multi Trip Vehicle Routing Problem where a time windows is associated with each customer and a *release date* is associated with each merchandise to be delivered at a certain client. The release date represents the moment the merchandise becomes available at the depot for final delivery.

The problem is relevant in city logistics context, where delivery systems based on city distribution centers (CDC) are studied. Trucks arrive at the CDC during the whole working day to deliver goods that are transferred to eco-friendly vehicles in charge of accomplish final deliveries to customers.

We propose a population-based algorithm for the problem based on giant tour representation of the chromosomes as well as a *split* procedure to obtain solutions from individuals.

Keywords: Multi trip; release dates; genetic algorithm; split.

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 minimizing routing costs and respecting capacity constraints on vehicles. The VRP represents a simplified problem that is usually far from the reality of freight distribution. To better represent real world problems, different aspects need to be taken into account leading to more complex problems, usually called *rich* (or *multi-attribute*) problems.

In this paper, we introduce a new variant of the VRP, the Multi Trip Vehicle Routing Problem with Time Windows and Release Dates. Our interest for this problem originates from mutualized distribution in cities, where external goods continuously arrive in a City Distribution Center (CDC) from where the last-mile delivery is operated

In this context, vehicle distribution from the CDC should be optimized on a daily basis. As vehicles have preferably small capacities and the fleet size should be minimized, vehicles will typically perform several trips along the day. This introduces the *multi trip* aspect.

Customers usually ask to be served within a certain time interval. Meeting these intervals is vital for the carrier: delays mean loosing reliability and trustworthiness and often means paying a penalty. Then, *time windows* should be considered and associated with each customer.

Finally, merchandise can be delivered to the CDC all day long. This means that they are not necessarily available at the CDC at the beginning of the planning horizon. Vehicle routes must then be designed such that no vehicle leaves the CDC before the goods it has to transport in its trip have arrived. The concept of *release date* is associated with each merchandise, indicating the time at which the merchandise is available at the CDC.

These attributes together lead to the Multi Trip Vehicle Routing Problem with Time Windows and Release Dates (MTVRPTW-R). It is noteworthy that the problem is *static* (or *off-line*) even if the merchandise continuously arrives to the CDC during the day. In fact, the release dates are supposed to be known before the working day starts.

Online variants can consider requests that become known at time intervals differently long before the respective release date. This aspect raises the issue of the value of the information. These variants, as well as stochastic variants, where the actual availability of the merchandise might differ from the release date, are left for future researches. To the best of our knowledge, this is the first time release dates are considered in a routing problem. Discussions with researchers from several software companies revealed that this issue, coupled with multiple trips and time windows, is actually very relevant in practice and models a problem that companies have to deal with (Grunert [9] and Kleff [16]).

The MTVRPTW-R implicitly models the dependence between different levels in multi-level distribution systems. The particular case of two-level distribution systems (called as well two-echelon or two-tier distribution systems, Hemmelmayr et al. [11], Crainic [8]) has recently largely been investigated by scholars. In these systems, final delivery trips depend on the time and on the CDC where vehicles operating in the first level unload goods. Information sharing and synchronization between the two levels of distribution are however assumed. On the contrary, the MTVRPTW-R models a situation where the first level of distribution cannot be controlled. The purpose of this paper is to contribute filling this gap introducing a new routing problem that takes into account this aspect as well as proposing an efficient procedure to solve it.

The paper is organized as follows. In Section 2 the problem is formally defined and characterized. Section 3 reviews related research. Section 4 describes a heuristic solution. Instance sets and results are presented in Section 5, while Section 6 concludes the paper.

2 Problem definition and notation

This section defines the problem (Section 2.1) and introduces the notation used in the rest of the paper (Section 2.2). Moreover, relations with the VRP with pickup and delivery are presented in Section 2.3. Finally, a characterization of problems with release dates is proposed in Section 2.4.

2.1 Problem definition

The MTVRPTW-R can be defined on a complete undirected graph $G = (V, E)$, where $V = \{0, \ldots, N\}$ is the set of vertices and $E = \{(i, j)|i, j \in V, i < j\}$ the set of edges. Vertex 0 represents the depot, where a fleet of M identical vehicles with capacity Q is based. Vertices $1, \ldots, N$ represent the customers. With each customer is associated a demand Q_i that needs to be delivered during a time window (TW) indicated by $[E_i, L_i]$. Service at customer i takes S_i and must start during the TW. Service time S_0 at the depot is usually called loading time. Arriving at customer location before E_i is allowed. Since the service cannot start earlier than E_i, the driver must wait. On the other side, late arrival at customer location is forbidden. Moreover, the quantity Q_i of product requested by customer i is available at the depot not earlier than R_i. R_i is called *release date*. For brevity, we will say R_i *is associated with customer* i, instead of R_i *is associated with the quantity* Q_i *requested by customer* i. It is possible to travel from i to j incurring in a travel time T_{ij} and covering a distance D_{ij}, $i, j = 0, \ldots, N$.

It is noteworthy that the release dates introduced here do not have the same implication as those considered in scheduling problems (called as well release or ready times, Błażewicz [4]). In that case, the release date is the earliest time at which job can be processed on a machine (Pinedo [25]). A vehicle needs to wait until all goods it has to carry are available at the depot, i.e., it cannot start the trip before the maximal release date associated with the customers it has to serve. On the other hand, a machine can

start processing available jobs without waiting for all to be ready.

A time horizon T_H is given and establishes the duration of the working day. It can be viewed as a TW associated with the depot. Thus, it is assumed that $[E_0, L_0] = [0, T_H]$, $Q_0 = 0$. Operations cannot start before E_0 and all vehicles must be back to the depot at time L_0.

The MTVRPTW-R calls for the determination of a set of trips and an assignment of each trip to a vehicle, such that the total travel distance is minimized and the following conditions are satisfied:

(a) each trip starts and ends at the depot;

(b) trips do not start earlier than $E_0 = 0$ and finish later than $L_0 = T_H$;

(c) no trips assigned to the same vehicle overlap;

(d) operations of each trip start not earlier than the greater R_i associated with customers assigned to the trip itself;

(e) each customer is visited by exactly one trip;

(f) service at customer i starts in the associated range $[E_i, L_i]$;

(g) the sum of the demands of the customers served by any trip does not exceed Q;

It is supposed that each customer i can be served by a return trip, i.e, $R_i + T_{0i} \leq L_i$, $\max\{E_i, R_i + T_{0i}\} + S_i + T_{i0} \leq T_H$ and $Q_i \leq Q$ (otherwise no feasible solution would exist).

2.2 Notation

The notation used in the rest of the paper is introduced here. The symbol σ will always indicate a trip. The capital Σ will indicate the set of trips assigned to a vehicle. It will be called *journey* in the following. A journey is formed by different trips. The symbol $\oplus$ is used to indicate the concatenation of paths (partial trips) or trips. For example $(v_1, \ldots, v_n) \oplus (w_1, \ldots, w_m)$ is the concatenation of two paths (that results in a trip if v_1 and w_m are the depot). $\sigma_1 \oplus \sigma_2$ means that trip σ_1 is performed right before σ_2 by the same vehicle.

The time a vehicle is available at the depot to perform a given trip σ, is indicated as T^σ. It is noteworthy that the vehicle cannot start trip σ before $\max\{T^\sigma, \max_{v \in \sigma} R_v\}$. Here and in the next sections, the symbol "$\in$" will be used to describe "belonging". For example $\sigma \in \Sigma$ means trip σ belongs to journey Σ, $v \in \sigma$ means customer v belongs to (i.e., it is served by) trip σ, and so on.

The service of a customer i will be called *feasible* if the vehicle arrives at its location before L_i, *infeasible* otherwise. A trip σ is called feasible if service at each customer in σ is feasible, infeasible otherwise. A journey is called feasible if it is composed by feasible trips, infeasible otherwise.

2.3 Relationship with the VRPPDTW

In the VRP with pickup and delivery (VRPPD) requests $i = 1, \ldots, N$ need to be picked up at defined locations $PICK_i$ before delivery takes place at location DEL_i. Precedence constraints impose pickups to be performed before deliveries. The VRPPD with time windows (VRPPDTW) asks pickups and deliveries to take place during a TW. The MTVRPTW-R can be reduced to an extension of the VRPPDTW where multiple trips are allowed. Given an instance of the MTVRPTW-R, N requests are defined such that $PICK_i$ is the depot for every request and DEL_i is the location of customer i. The pickup TW associated with request i is set to $[R_i, T_H]$, the delivery TW is set to $[E_i, L_i]$. Travelling from the depot to a pickup location and between pickup locations does not require time, i.e., $T_{0PICK_i} = 0$, $T_{PICK_i 0} = 0$ and $T_{PICK_i PICK_j} = 0$. On the other hand $T_{PICK_i DEL_j} = T_{0j}$ and $T_{DEL_i DEL_j} = T_{ij}$. Distances are managed similarly.

2.4 Problem characterization

Larsen [17] and Larsen et al. [18] introduce some parameters to characterize problems in the context of the Dynamic VRP (DVRP). Even if there is no exact correspondence between the DVRP and the MTVRPTW-R, these problems share similarities. In the DVRP, a request becomes known at a certain time called *disclosure time* (Pillac et al. [24]), while in the MTVRPTW-R, merchandise becomes available at a certain moment that we call *release date*. The main difference is that all the release dates are known at the beginning of the working day, that makes the MTVRPTW-R a static problem. On the other side, requests in the DVRP context become known during operations.

We introduce the *rigidity* of a system (that correspond to the degree of dynamism for the DVRP) as

$$r = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{L_i - R_i}{T_H} \right). \tag{1}$$

Since $R_i \leq L_i$, $r \in [0, 1]$. It represents how averagely close are the R_i to the corresponding L_i. In particular, $r = 0$ when $R_i = 0$ and $L_i = T_H$ for all customers i. On the other side $r = 1$ when $R_i = L_i$ for all customers i. A considerable amount of release dates R_i close to the corresponding L_i thus makes the plan more rigid than having the release dates R_i far in time from L_i.

It can be noticed that, when $R_i = 0$ for all customers i, the rigidity r can be strictly positive since it depends on the relationship between L_i and T_H. We introduce another parameter that takes into account only the relationship between R_i and L_i. We define the *instance tightness* as

$$tigh = \frac{1}{N} \sum_{i=1}^{N} \frac{R_i}{L_i}. \tag{2}$$

The parameter $tigh$ equals zero when $R_i = 0$ for all customers i and equals one when $R_i = L_i$ for all customers i. Practically, r and $tigh$ are strictly less than 1: all $R_i = L_i$ would make the problem infeasible as long as travel times between the depot and the customers are not null.

3 Literature Review

To the best of our knowledge, there is no routing problem considering release dates. On the other side, the Multi Trip VRP with Time Windows (MTVRPTW) is an extension of the well-known Vehicle Routing Problem with Time Windows (VRPTW) where customers must be served during a time interval called *time window* (Bräysy and Gendreau [5, 6]) and the Multi Trip VRP, that allows vehicles to perform several trips during the working day (Cattaruzza et al. [7], Olivera and Vera [23], Taillard et al. [32]).

The MTVRP is a $\mathcal{NP}$-hard problem (Olivera and Viera [23]). This makes the MTVRPTW a $\mathcal{NP}$-hard problem (the MTVRP can be reduced to the MTVRPTW associating a TW equal to $[0, T_H]$ with each customer). Finally, the MTVRPTW-R is $\mathcal{NP}$-hard since the MTVRPTW trivially reduces to the MTVRPTW-R setting all the release dates to zero.

Despite its practical interest, the literature on the MTVRPTW is pretty scarce and most of the existing papers propose exact methods. Azi et al. [1] propose an exact algorithm for solving the single vehicle MTVRPTW. The solution approach exploits an elementary shortest path algorithm with resource constraints. In the first phase all non dominated paths are calculated. Then the shortest path algorithm is applied to a modified graph. Each node represents a non-dominated trip and two nodes are connected by an arc when it is possible to serve the two trips consecutively and they do not serve common customers. Solomon's instances are used with different values of time horizon. 16 instances out of 54 with 100 customers are solved to optimality.

Azi et al. [2] address the MTVRPTW where trips are constrained by a limited duration and service at all customers is not mandatory. A column generation approach embedded within a branch-and-price algorithm is proposed. A set packing formulation is given for the master problem and each column represents a working day. Since each pricing problem is an elementary shortest path with resource constraints, a similar approach to the one proposed in Azi et al. [1] is applied. As in Azi et al. [1], Solomon's instances are considered. Due to the limitations of the algorithm, the authors focus on instances formed by the first 25 or 40 customers of each Solomon's instance. The algorithm can also solve few instances of size 50. Hernandez et al. [13] use a similar approach. A set covering formulation is given for the problem and each column represents a trip instead of a working day. With this method applied on the same instances proposed in Azi et al. [2] optimal solutions are found for a majority of instances with size up to 50. Macedo et al. [19] propose a minimum flow model where variable represent feasible trips. Optimal solutions are found for a majority of instances with size up to 50 in 2 hours of computation time. In all the previous works, the full set of feasible trips is generated. This is practicable due to the presence of time windows and trip duration constraint that limit the cardinality of the set.

Hernandez et al. [12] propose the only exact method on the MTVRPTW. A branch and price algorithm is proposed and instances up to 50 customers and 4 vehicles can be solved.

Battarra et al. [3] study an extension of the MTVRPTW where products are clustered in different commodities that cannot be transported in the same vehicle. They first generate a set of feasible trips considering each commodity independently. Then, these trips are assigned to vehicles in order to obtain a solution.

4 Method

This section describes the hybrid genetic algorithm $\mathcal{A}^{CAF}$ we developed for the MTVRPTW-R. In genetic algorithms (GA), a set of *chromosomes* forms a population that evolves across generations until termination criteria are met. New individuals (*children*) are generated from those in the current population, called *parents*, by *crossover* and *mutation* operators. GAs turned out to be highly efficient heuristic methods to face different problems, mainly because children generation allows to explore new zones (children differ from parents) that are promising (children keep good parents' characteristics). The papers of Prins [26], Vidal et al. [33] and Cattaruzza et al. [7] are three examples of efficient GA (respectively for the VRP; the multi depot VRP and the periodic VRP; and for the multi trip VRP) in the VRP field.

The principles of genetic procedure were first formalized by Holland [14]. The interested reader is referred to Reeves [28], Moscato and Cotta [20] and Neri and Cotta [22] for an overview of population based procedures.

4.1 Algorithm outline

This section outlines our $\mathcal{A}^{CAF}$ for the MTVRPTW-R. Initially, a population Π of π chromosomes is generated. Chromosomes are sequences of client nodes without trip delimiters. Each chromosome is evaluated with an adaptation of the Split procedure developed by Prins [26], that we call *AdSplit* (Section 4.4). *AdSplit* obtains a MTVRPTW-R solution ξ from a chromosome Ψ. The fitness of chromosome Ψ is then defined as the cost of the solution ξ.

All individuals are improved by means of a local search (LS) procedure called as well *education* phase (Section 4.3). At each iteration two chromosomes Ψ_{P_1} and Ψ_{P_2} are selected using the classical binary tournament procedure. Two children Ψ_{C_1} and Ψ_{C_2} are obtained crossing parents with the order crossover procedure and one (Ψ_C) is randomly selected between them. Ψ_C undergoes *AdSplit* for evaluation and LS for education. It is then inserted in Π.

After μ children are inserted in Π, π chromosomes are selected based on their fitness (representing

their quality) and diversification contribution to the population itself, while the other μ are eliminated (survivor procedure, Section 4.5). A sketch of the method is given in Algorithm 1.

Algorithm 1 $\mathcal{A}^{CAF}$ outline

```
Initialize population
while Termination criteria are not met do
    Select parent chromosomes Ψ_{P_1} and Ψ_{P_2}
    Generate a child Ψ_C
    Educate Ψ_C
    Insert Ψ_C in the population
    if Dimension of the population exceeds a given size then
        Select survivors
    end if
end while
```

4.2 Solution representation and search space

A chromosome is a sequence (permutation) $\Psi = (\Psi_1, \ldots, \Psi_N)$ of the N client nodes, without trip delimiters. Ψ can be viewed as a TSP solution that has to be turned into a feasible MTVRPTW-R solution by splitting the chromosome (inserting trip delimiters and assigning trips to vehicles). Ψ is usually called a *giant tour*. From a giant tour Ψ, different MTVRPTW-R solutions can be constructed depending on the way Ψ is split.

During the search phase, overload and TW violations are allowed and penalized in the fitness function. Two penalization factors are needed: θ for TW violation and λ for load infeasibility.

A procedure *AdSplit* (explained in Section 4.4) is used to get a MTVRPTW-R solution ξ from Ψ. The following notation is introduced: $D_\Sigma(\xi)$ and $TW_\Sigma(\xi)$ are respectively the traveled distance and the TW violation of journey Σ in solution ξ. $L_\sigma(\xi)$ is the load of trip σ. The fitness $F(\Psi)$ of the chromosome Ψ is the cost $c(\xi)$ of the solution ξ found by *AdSplit* and it is defined as

$$F(\Psi) = c(\xi) = \sum_{\Sigma=1}^{M} D_\Sigma(\xi) + \theta \sum_{\Sigma=1}^{M} TW_\Sigma(\xi) + \lambda \sum_{\Sigma=1}^{M} \sum_{\sigma \in \Sigma} \max\{0, L_\sigma(\xi) - Q\}. \tag{3}$$

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

In practice a chromosome is split in order to obtain a solution and then it possibly undergoes LS for improvement. It would then be natural to continue describing the solution method with the *AdSplit* procedure. However, *AdSplit* takes advantages of aspects in the LS. Thus, we will start presenting the latter.

4.3 Local search

This section presents the local search (LS) procedure embedded in $\mathcal{A}^{CAF}$. First, an efficient scheme to manage TW violations is presented (Section 4.3.1) based on Vidal et al. [34] work. In Section 4.3.2 peculiar characteristics of our problem are introduced, and Section 4.3.3 describes the general LS procedure.

4.3.1 Local Search for VRP with TW

LS in presence of TW becomes more complicated than in the classic VRP. Feasibility checks and routing cost variations cannot be straightforwardly calculated in constant time. Savelsbergh [29] proposes a scheme

to check feasibility and to calculate cost variation for series of k-opt moves. Nagata et al. [21] propose a new scheme to evaluate TW violations. Roughly speaking, when a vehicle arrives late at the customer location, it is allowed to drive back in time in order to meet the TW and a penalization proportional to the TW violation is introduced in the objective function. They also propose formulas to evaluate in constant time inter-route moves as relocation, exchange and 2-opt (inter-route 2-opt is usually called 2-opt*). [1].

Vidal et al. [34] generalize this scheme for a large class of routing problems with TW. Each move is seen as a concatenation of paths. In particular, given a path ρ, the quantities $T(\rho)$, $TW(\rho)$, $E(\rho)$, $L(\rho)$, $D(\rho)$ and $Q(\rho)$ are introduced. $T(\rho)$ and $TW(\rho)$ are respectively the minimum duration and the minimum penalization (called as well *time warp*) of ρ. $E(\rho)$ and $L(\rho)$ are the earliest and the latest date service can start at the first customer of ρ (that can be the depot) allowing minimum duration and TW violation. $D(\rho)$ is the travelled distance while $Q(\rho)$ is the cumulative demand of served customer. We will call these quantities *features*. For a path made by a single customer i, features are initialized as follows: $T(i) = S_i$, $TW(i) = 0$, $E(i) = E_i$, $L(i) = L_i$, $D(i) = 0$, $Q(i) = Q_i$. Given two paths ρ_1 and ρ_2, the following relations hold (see Vidal et al. [34] for formal proofs):

$$T(\rho_1 \oplus \rho_2) = T(\rho_2) + T(\rho_1) + T_{v^1_{n_1}, v^2_1} + \Delta_{WT}; \tag{4}$$
$$TW(\rho_1 \oplus \rho_2) = TW(\rho_1) + TW(\rho_2) + \Delta_{TW}; \tag{5}$$
$$E(\rho_1 \oplus \rho_2) = \max\{E(\rho_2) - \Delta, E(\rho_1)\} - \Delta_{WT}; \tag{6}$$
$$L(\rho_1 \oplus \rho_2) = \min\{L(\rho_2) - \Delta, L(\rho_1)\} + \Delta_{TW}; \tag{7}$$
$$D(\rho_1 \oplus \rho_2) = D(\rho_1) + D(\rho_2); \tag{8}$$
$$Q(\rho_1 \oplus \rho_2) = Q(\rho_1) + Q(\rho_2); \tag{9}$$

where

$$\Delta = T(\rho_1) - TW(\rho_1) + T_{v^1_{n_1}, v^2_1};$$
$$\Delta_{WT} = \max\{E(\rho_2) - \Delta - L(\rho_1), 0\};$$
$$\Delta_{TW} = \max\{E(\rho_1) + \Delta - L(\rho_2), 0\}.$$

Equations (4)–(9) allow evaluating classical LS moves in constant time. For example, relocate customer v_i from trip $\sigma_1 = (0, v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_{n_1}, 0)$ to trip $\sigma_2 = (0, w_1, \ldots, w_j, w_{j+1}, \ldots, w_{n_2}, 0)$ between customers w_j and w_{j+1} can be evaluated applying Equations (4)–(9) to $(0, v_1, \ldots, v_{i-1}) \oplus (v_{i+1}, \ldots, v_{n_1}, 0)$ and $(0, w_1, \ldots, w_j) \oplus (v_i) \oplus (w_{j+1}, \ldots, w_{n_2}, 0)$.

Features defined in Equations (4)–(9) need to be available for all the paths and their reverse present in the current solution. Reverse paths come into play, for example, in the evaluation of 2-opt moves. Moreover, these quantities need to be updated each time a move is implemented. Straightforward update takes $O(N^2)$, but it can be done in $O(N^{8/7})$ using speed up techniques proposed by Irnich [15].

We use this approach (except for the little modification introduced by the release dates, Section 4.3.2) to evaluate *local* variations due to a move. By *local* we mean a cost variation that occurs in the trips affected by the move itself. It is noteworthy that this variation can affect successive trips in the same journey (Section 4.3.2).

4.3.2 Local search: introduction of release dates and application to the multi trip case

The MTVRPTW-R has two more attributes to consider compared to the VRPTW: vehicles can perform several trips, and merchandise can be non-available at the depot at the beginning of the horizon.

[1] The formula proposed for inter-route relocations provides incorrect results in particular cases and it has been corrected by Schneider et al. [30].

We introduce a new feature $R(\rho)$ as the greatest release date of customers served in ρ. We define $R(i) = R_i$ and Equation (10) holds

$$R(\rho_1 \oplus \rho_2) = \max\{R(\rho_1), R(\rho_2)\}. \tag{10}$$

Then, given two paths ρ_1 and ρ_2, the quantity $R(\rho_1 \oplus \rho_2)$ can be calculated in constant time from values $R(\rho_1)$ and $R(\rho_2)$.

It is observed that in VRP it is always possible to start a new trip σ at time t, with $E(\sigma) \leq t \leq L(\sigma)$, because all the vehicles are available at the depot at $t = 0$ and are assigned to only one trip. Then, the minimum travelling duration time $T(\sigma)$ and the minimum TW violation $TW(\sigma)$ can always be obtained. On the other side, this cannot be possible when vehicles are allowed to perform more than one trip or in the presence of release dates. In the first case it can be $L(\sigma) < T^{\sigma}$ because of previous trips, in the second case, it can be that $L(\sigma) < R(\sigma)$ (that means trip σ is infeasible). We, then, need to calculate $T(\sigma)$ and $TW(\sigma)$ based on the effective time t a vehicle leaves the depot to perform σ. To do that, we can use the following relations proved by Vidal et al. [34] that provide the value of these quantities as functions of the starting time t:

$$T(\rho)(t) = T(\rho) + \max\{0, E(\rho) - t\}; \tag{11}$$

$$TW(\rho)(t) = TW(\rho) + \max\{0, t - L(\rho)\}. \tag{12}$$

Finally, the time T^{σ_i} a vehicle is available at the depot to perform trip σ_i can be recursively calculated as follows

$$T^{\sigma_1} = 0; \tag{13}$$

$$T^{\sigma_{i+1}} = T(\sigma_i)(\max\{R(\sigma_i), T^{\sigma_i}\}) - TW(\sigma_i)(\max\{R(\sigma_i), T^{\sigma_i}\}). \tag{14}$$

The Figure 1 introduces an example (data are given in tables, travel times and distances coincide) that illustrates a consequence of using Nagata et al. [21] penalization scheme for TW violations in the multi-trip context. In particular Figure 1 depicts a journey formed by four trips: $\sigma_1 = (v_0, v_5, v_3, v_0)$, $\sigma_1 = (v_0, v_1, v_0)$, $\sigma_3 = (v_0, v_4, v_0)$ and $\sigma_4 = (v_0, v_2, v_0)$ where v_0 represents the depot. It can be noticed that $115 = T^{\sigma_4} < T^{\sigma_3} = 125$ and $95 = T^{\sigma_5} < T^{\sigma_4}$. This means that the vehicle can be available at the depot to perform trip σ_{i+1} before the time it was available to perform trip σ_i. Indeed, it travelled back in time and it is the consequence of a deep use of the time warp, i.e., a deep time window violation.

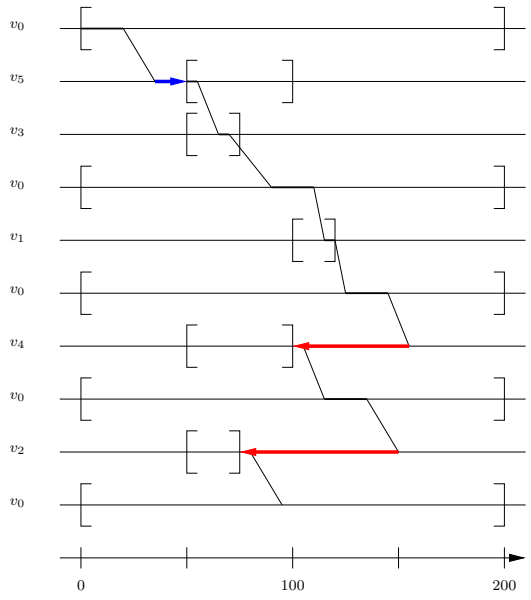
One needs also to observe that the effect of a move on trip $\sigma_i \in \Sigma$ can modify $T^{\sigma_{i+1}}$, that in turn can modify $T(\sigma_{i+1})$ and $TW(\sigma_{i+1})$. Hence, the effect of the move on trip σ_i can be propagated to the following trips.

When loading times at the depot are constant and not trip-dependent, evaluation of the TW violation can be done in constant time, considering a journey as a unique segment with multiple visit at the depot. Conversely, the calculation requires $O(\Sigma_{\max})$, where $\Sigma_{\max}$ indicates the maximum number of trips among all journeys.

4.3.3 Local search: general scheme and penalty adaptation

We consider the move 2-opt and the move ex_ρ that exchanges two segments ρ_1 and ρ_2 of successive customers, where ρ_1 and ρ_2 are respectively subsequences of trips σ_1 and σ_2. Let v_1 and v_2 be two customers. In particular, it can be observed that if:

- $\rho_1 = v_1$ and $\rho_2 = \emptyset$, ex_ρ is the classical relocate;
- $\rho_1 = v_1$ and $\rho_2 = v_2$, ex_ρ is the classical exchange (or swap);



	S_i	Q_i	E_i	L_i	R_i
v_0	20	0	0	200	0
v_1	5	20	100	120	60
v_2	5	20	50	75	0
v_3	5	20	50	75	0
v_4	5	20	50	100	60
v_5	5	20	50	100	0

Distance matrix						
	v_0	v_1	v_2	v_3	v_4	v_5
v_0	0	5	15	20	10	15
v_1	5	0	20	20	15	15
v_2	15	20	0	40	20	30
v_3	20	20	40	0	30	10
v_4	10	15	20	30	0	20
v_5	15	15	30	10	20	0

Figure 1: The last two trips arrive at the depot before they have left it

- $\rho_1 = (v_1, \ldots, 0)$ and $\rho_2 = (v_2, \ldots, 0)$, or $\rho_1 = (0, \ldots, v_1)$ and $\rho_2 = (0, \ldots, v_2)$, and $\sigma_1 \neq \sigma_2$, ex_ρ is the classical 2-opt*;
- $\rho_1 = \sigma_1$ and $\rho_2 = \emptyset$, ex_ρ relocates a full trip;
- $\rho_1 = \sigma_1$ and $\rho_2 = \sigma_2$, ex_ρ exchanges trips.

The last two moves, take into account the multi-trip aspect. Relocations and swaps of trips are inter-vehicle (resp. intra-vehicle) moves if σ_1 and σ_2 belong to a different (resp. the same) journey. No limitation on the segment size is considered. Different neighbour structures considering a maximum segment size could speed up the search against a payback due to the possible lower quality of the obtained solutions.

Initially, a neighbourhood defined by one of the listed move is randomly chosen and deeply explored. The best improving move is then executed, if it exists. In this case, another neighbourhood is randomly chosen. The LS terminates when all the neighborhoods are fully explored without finding any improving move.

When the population reaches the dimension of $\pi + \mu$, the values of each penalty factor $P = \theta, \lambda$ is adjusted as follows:

$$P = \begin{cases} 1.25 \times P & \text{if } \zeta_P > \zeta^+_{ref}; \\ \max\{1, 0.85 \times P\} & \text{if } \zeta_P < \zeta^-_{ref}, \end{cases}$$

where ζ_P is the percentage of infeasible chromosomes over the last μ individuals generated with respect to time window violation ($P = \theta$) or capacity violation ($P = \lambda$), while ζ^+_{ref} and ζ^-_{ref} are reference values.

4.4 A Split algorithm for the multi trip problem with TW and release dates

The split procedure, indicated with *AdSplit*, is an adaptation of the procedure proposed by Prins [26]. It turns a permutation of the N customers into a solution for the MTVRPTW-R. It works on an acyclic graph H whose nodes represent customers, and arcs represent trips. The graph construction is illustrated in Section 4.4.1, while Section 4.4.2 presents the *AdSplit* procedure.

4.4.1 Auxiliary graph construction

The auxiliary graph $H = (V', A')$ is defined with $N + 1$ nodes indexed from 0 to N. Node i represents customer Ψ_i in the chromosome $\Psi = (\Psi_1, \ldots, \Psi_N)$. Arc (i, j), $i < j$, represents trip σ_{i+1}^j serving customers from Ψ_{i+1} to Ψ_j in this order, i.e., $\sigma_{i+1}^j = (0, \Psi_{i+1}, \ldots, \Psi_j, 0)$. Cost c_{ij} of arc (i, j) is given by the following equation

$$c_{ij} = D(\sigma_{i+1}^j) + \theta TW(\sigma_{i+1}^j) + \lambda \max\{0, Q(\sigma_{i+1}^j) - Q\},$$

that is the sum of the travelled distance in trip σ_{i+1}^j plus the penalized TW and capacity violations. Figure 2 depicts graph H for the example in Figure 1, where $\Psi = (1, 2, 3, 4, 5)$. Cost of arc $(0, 1)$ is

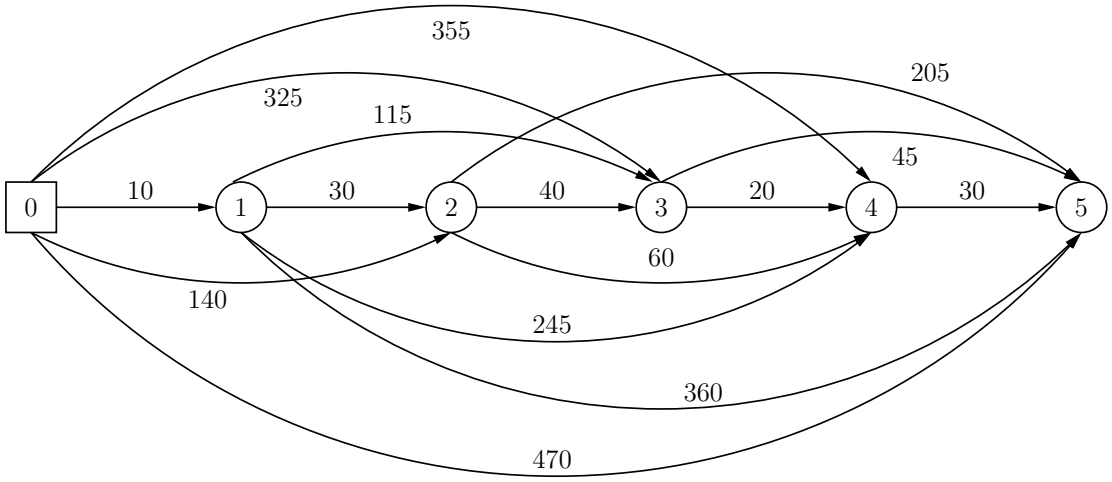


Figure 2: The auxiliary graph for chromosome $\Psi = (1, 2, 3, 4, 5)$, data as in Figure 1, $Q = 60$ and $\theta = \lambda = 2$

twice the travelling distance needed to go from the depot to customer 1, i.e., 10. Cost of arc (0,2) is the travelling distance needed to reach customer 1 from the depot, then visit customer 2 and to conclude the trip at the depot, i.e., 40. The penalization $50 \cdot \theta$ is added since the vehicle arrives at customer 2 at 125 and its time window closes at 75. Then, for $\theta = 2$, arc $(0, 2)$ costs 140. Other costs are computed similarly.

It is noteworthy that the arc cost does not take into account the position of the trip in the journey, but it is the (penalized) cost of the trip when it is performed at $t = 0$. Therefore, the (contingent) TW violation due to later departure is not taken into account.

4.4.2 Assignment of trips to vehicles

In the MTVRPTW-R context in particular and in the MTVRP context in general, more than one trip can be assigned to the same vehicle. TW penalization deeply depends on the time trips leave the depot and this aspect cannot be considered by the constant costs associated with arcs on H.

Due to the correspondence between arcs and trips in H and to simplify the presentation, in this section we will indifferently refer to arc or trip. Consequently, the assignment of arcs to a vehicle, will mean that the trips represented by these arcs are assigned to this vehicle.

Each path that goes from node 0 to node N, represents a set of trips, that need to be assigned to vehicles to form a solution. Arcs are assigned to a vehicle in the order they appear in the path. For each arc, M assignments (one for each vehicle) are possible. In each case, the trip is positioned after the trips already assigned to the vehicle. Hence, the order of trips in vehicles meets the order in Ψ.

The aim of the *AdSplit* procedure is to find the path whose optimal assignment of trips to vehicles results in the best solution. It can be shown that the problem solved with the *AdSplit* procedure in $\mathcal{NP}$-hard.

Let us consider the problem $SPLIT$ that consists in finding the optimal solution for an auxiliary graph H (constructed as explained in Section 4.4.1) defined by an instance I of the MTVRPTW-R and a

permutation Ψ of customers in I. The objective of $SPLIT$ is to find the optimal solution ξ_Ψ associated with Ψ on graph H. In what follows we analyze the complexity of the $SPLIT$ problem.

Proposition 1. *The problem $SPLIT$ is $\mathcal{NP}$-Hard.*

Proof. We perform a reduction from the Bin Packing Problem (BPP). The decision version of the BPP is defined given M^{BPP} bins of size Q^{BPP} and N^{BPP} objects of size $Q_i^{BPP}, i = 1, \ldots, N^{BPP}$. We need to answer the question: can the N^{BPP} objects be inserted in the M^{BPP} available bins respecting capacity constraints?

The key idea of the reduction is to define an instance of the $SPLIT$ problem such that each arc (i, j) corresponds to the selection of objects $\{i + 1, \ldots, j\}$ of the BPP.

Given an instance I^{BPP} of the BPP, an instance of the MTVRPTW-R problem is defined with the following parameters: $N = N^{BPP}$, $T_{ij} = Q_j^{BPP}$ for all $i, j = 1, \ldots, N$, and $T_{0i} = Q_i^{BPP}$, $T_{i0} = 0$, $[E_i, L_i] = \left[0, Q^{BPP}\right]$, $R_i = 0$, for all $i = 1, \ldots, N$ and $Q = +\infty$. We call I^{SPLIT} the instance of the decision version of problem $SPLIT$ obtained when applying *AdSplit* to the chromosome $(1, \ldots, N)$ for this instance.

The graph H associated with the I^{SPLIT} instance is constructed such that the cost on each arc (i, j) is given by $c_{ij} = \sum_{k=i+1}^{j} Q_k$ for all $i, j = 0, \ldots, N$, $i < j$. We can notice that all paths from 0 to N in this graph have the same cost ($\sum_{k=1}^{N} Q_k$). From the definition of I^{SPLIT}, it is obvious that the instance I^{BPP} can be polynomially transformed into an I^{SPLIT} instance.

We show that the answer for instance I^{BPP} is positive if and only if there exists a feasible solution of the $SPLIT$ problem that uses at most M^{BPP} vehicles.

If I^{BPP} is a positive instance, then there exists an assignment of objects to bins such that the number of used bins is lower than or equal to M^{BPP}. Considering the path in H that corresponds to all the arcs $(i - 1, i), i = 1, \ldots, N$ and assigning trips $(0, i, 0)$ to the k^{th} vehicle if the i^{th} object is into the k^{th} bin, constructs a feasible solution for I^{SPLIT}.

On the other side let suppose that I^{SPLIT} is a positive instance. There exists a path ρ from 0 to N and an assignment of arcs to the M^{BPP} vehicles such that the sum of the duration of the trips assigned to a vehicle does not exceed the time limit Q^{BPP}. A solution for the I^{BPP} instance is constructed assigning object i to bin k if customer i is assigned to the k^{th} vehicle in ρ. The number of used bins is lower than or equal to M^{BPP} and their capacities are satisfied, which concludes the proof.

□

We propose a labelling procedure that selects arcs on H and assigns them to vehicles, in order to obtain the solution with minimum cost with respect to Equation (3).

Labels are associated with nodes in the graph. Each label associated with node i represents a partial path that goes from node 0 to node i in H and, then, a partial solution that serves all customers $\Psi_1, \ldots, \Psi_i$. Trips of this partial solution are represented by the arcs in the corresponding partial path.

Each label $\mathcal{L}$ has $M + 3$ fields. Each field j, with $j = 1, \ldots, M$ records the availability time $T_j(\mathcal{L})$ of vehicle j, namely, the time the vehicle is available at the depot for starting the next service trip. Availability times are stored in decreasing order to better take advantage of the dominance rules introduced in the following. The $(M + 1)^{\text{th}}$ field memorizes the total load infeasibility, and the $(M + 2)^{\text{th}}$ the predecessor node. The last field stores the cost $c(\mathcal{L})$ of the partial solution represented by $\mathcal{L}$. When extending a label, M new labels are constructed, one for each possible allocation of the new trip to a vehicle. *Extending* a label $\mathcal{L}_i$ associated with node i through arc (i, j) means assigning trip $(0, \Psi_{i+1}, \ldots, \Psi_j, 0)$ to a certain vehicle. Identical labels associated with the same node are discarded.

Graph H is implicitly generated. Arc and solution costs are computed using relations introduced in Section 4.3. In particular, the cost of an arc $(i, \ldots, j+1)$ is calculated concatenating path $(0, \Psi_{i+1}, \ldots, \Psi_j)$ with path $(\Psi_{j+1}, 0)$, while label costs are updated using Relations (11)–(14) since vehicle availability times are stored in labels.

1				2				3				4				5			
cost	label	p	dom	cost	label	p	dom	cost	label	p	dom	cost	label	p	dom	cost	label	p	dom
10	(110,000)	0	no	40	(110, 70)	1	no	125	(110, 100)	1	no	190	(115, 110)	2	no	265	(120, 110)	2	no
				140	(95, 0)	0	no	180	(95, 75)	2	no	210	(115, 95)	3	no	285	(120, 95)	3	no
								230	(100, 70)	2	yes	250	(115, 75)	3	yes	300	(120, 115)	4	yes
								275	(100, 0)	0	no	250	(115, 100)	3	yes	300	(120, 100)	3	yes
												270	(115, 70)	2	yes	325	(120, 75)	3	yes
												295	(105, 100)	3	yes	345	(120, 70)	2	yes
												340	(115, 0)	2	no	370	(115, 70)	4	yes
												360	(105, 100)	3	yes	395	(120, 105)	4	yes
																415	(120, 0)	2	no

Table 1: Labels generated by *AdSplit*

The label $\mathcal{L}$ with minimum cost $c(\mathcal{L})$ associated with node N is selected and the related solution is constructed (going backwardly through the graph).

To speed up the procedure, dominated labels, accordingly with the following *dominance rule*, are discarded.

Dominance Rule 1 (Strong)**.** *Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be two labels associated with the same node i. $\mathcal{L}^1$* strongly *dominates $\mathcal{L}^2$ if and only if*

$$c(\mathcal{L}^1) + \theta \sum_{j=1}^{M} \delta_j(\mathcal{L}^1, \mathcal{L}^2) \leq c(\mathcal{L}^2); \tag{15}$$

$$\delta_j(\mathcal{L}^1, \mathcal{L}^2) = \max\{0, T_j(\mathcal{L}^1) - T_j(\mathcal{L}^2)\}.$$

Roughly speaking, $\delta_j(\mathcal{L}^1, \mathcal{L}^2)$ represents the maximal additional penalization that can be introduce in the partial solution represented by $\mathcal{L}^1$ compared to the one represented by $\mathcal{L}^2$. If Inequality (15) holds, $\mathcal{L}^2$ cannot be extended in a better way than $\mathcal{L}^1$, and it is eliminated.

Table 1 shows all the labels generated by *AdSplit* when applied on the graph H depicted in Figure 2. The number of vehicles M is set to 2. Column *dom* indicates if the respective label is dominated in the sense of Dominance Rule 1. Only 3 labels associated with node 5 are non-dominated.

Label $(120, 110)$ with cost $c = 265$ associated with node 5 corresponds to the solution made up of three trips: $(0, 1, 0)$ assigned to one vehicle and trips $(0, 2, 0)$ and $(0, 3, 4, 5, 0)$ performed by the other vehicle (in this order). Travelled distance is 125, while late arrival at customers $3, 4$ and 5 introduces a total TW violation of $70 \cdot \theta$.

Preliminary tests showed that a huge number of labels needs to be treated, which does not appear to be viable in the heuristic context (see Section 5.3 for details). For this reason a heuristic version of the Dominance Rule 1 is introduced as follows:

Dominance Rule 2 (Weak)**.** *Label $\mathcal{L}^1$* weakly *dominates $\mathcal{L}^2$ if and only if*

$$c(\mathcal{L}^1) + \theta \sum_{j=1}^{M} \delta_j(\mathcal{L}^1, \mathcal{L}^2) \leq \gamma c(\mathcal{L}^2), \tag{16}$$

$$c(\mathcal{L}^1) \leq c(\mathcal{L}^2). \tag{17}$$

where $\gamma \geq 1$.

For $\gamma = 1$, the weak dominance rule is equivalent to the strong version. When $\gamma > 1$, Inequality (16) is easier to be satisfied and a bigger number of labels can be eliminated. Condition (17) is added because when $\gamma > 1$ label $\mathcal{L}^2$ can be dominated by label $\mathcal{L}^1$ even if $c(\mathcal{L}^2) < c(\mathcal{L}^1)$. Using the weak relation one expects the solution to be obtained quicker. On the other side the best decomposition of the chromosome can be missed.

The value of γ is dynamically adapted during the process according to the number of labels associated with each node. Precisely the following scheme is adopted:

$$\gamma = \begin{cases} \gamma + \frac{|\mathcal{L}_i|}{1000\mathcal{L}_{\text{threshold}}} & \text{if } |\mathcal{L}_i| > \mathcal{L}_{\text{threshold}} \\ \gamma - \frac{\mathcal{L}_{\text{threshold}}}{1000|\mathcal{L}_i|} & \text{if } |\mathcal{L}_i| < \mathcal{L}_{\text{threshold}} \end{cases} \tag{18}$$

where $|\mathcal{L}_i|$ is the number of labels associated with node i and $\mathcal{L}_{\text{threshold}}$ is a threshold parameter that indicates the number of labels that should be kept associated with each node. If after Relation (18) is applied, γ is lower than 1, it is set back to 1.

It is worth noting that the smaller $\mathcal{L}_{\text{threshold}}$ is, the quicker *AdSplit* is. On the other side, the bigger $\mathcal{L}_{\text{threshold}}$ is, the better the quality of the solution obtained is.

4.5 Survivor strategy

When the population Π contains $\pi + \mu$ chromosomes, the *survivor* procedure is launched. It selects μ chromosomes based on their quality and their diversification contribution to the population as suggested by Vidal et al. [34, 33]. Diversity contribution $f(\Psi)$ is set as the average distance between a selected chromosome Ψ and its n_c closest neighbours in Π (forming set N_c). Distance $D(\Psi, \Psi_1)$ between chromosomes Ψ and Ψ_1 is measured by the *broken pair* distance that is the number of pairs of adjacent customers in Ψ that are broken in Ψ_1 (Prins [27]). Formally we have:

$$f(\Psi) = \frac{1}{n_c} \sum_{\Psi_1 \in N_c} D(\Psi, \Psi_1), \tag{19}$$

A biased fitness $bF(\cdot)$ is calculated for each chromosome as follows:

$$bF(\Psi) = r_F(\Psi) + (1 - \frac{n_e}{|\Pi|}) r_f(\Psi); \tag{20}$$

where $r_F(\Psi)$ and $r_f(\Psi)$ are the ranks of chromosome Ψ calculated based on fitness F and function f defined in Equation (19) respectively, and n_e is a parameter that ensures elitism properties during selection (see Vidal et al. [33] for a formal proof).

5 Computation results

In this section we present the results obtained with our procedure $\mathcal{A}^{CAF}$. We start introducing a set of instances (Section 5.1) for benchmark purposes: due to the novelty of the problem, no instances are available in the literature. In Sections 5.2 we present the values of the parameters involved in the $\mathcal{A}^{CAF}$, determined using a meta-tuning procedure. Results obtained on the new set of instances are presented in Section 5.4, while in Section 5.5 results obtained by $\mathcal{A}^{CAF}$ on instances proposed by Hernandez et al. [12] for the MTVRPTW are given. $\mathcal{A}^{CAF}$ is coded in C++ and compiled with Visual Studio 2010. All the experiments are run on a Intel Xeon W3550 3.07GHz with a RAM of 12 Gb.

5.1 Instances for the MTVRPTW-R

In this section we describe how we generate a set of instances for the MTVRPTW-R. We use the instances introduced by Solomon [31] for the VRPTW as base instances and adapt them to the MTVRPTW-R case. In Solomon [31] six groups of instances are generated, named R1, C1, RC1, R2, C2, RC2. Groups R1 and R2 have customers randomly located in a region, while they are clustered in groups C1 and C2. RC1 and RC2 instances contain a mix of randomly located and clustered customers. The time horizon is shorter in instances of groups C1, R1, RC1 than in instances of C2, R2, RC2. There are 56 instances in total.

Depot location as well as customer locations, demands, time windows and service times are set as in the original Solomon's instances. Release dates are calculated in three steps, based on the Algorithm 2 explained in the following, and given a rigidity parameter r.

In the first step (lines 1–7 of Algorithm 2) release dates $R^{(2)}$ are calculated based on Equation (21)

Algorithm 2 Instances creation

for all $i = 1, \ldots, N$ **do**
  if $L_i + T_H(r-1) \geq 0$ **then**
    $R_i = \lfloor L_i + T_H(r-1) \rfloor$
  else
    $R_i = 0$
  end if
end for
for all $i = 1, \ldots, N$ **do**
  if $R_i + T_{0i} > L_i$ **then**
    $R_i = 0$
  end if
end for
Initialize Λ with customers with strictly positive release date and ranked with respect to non-decreasing release dates: $u < v \Rightarrow R_u \leq R_v$
$\Gamma = \emptyset; \quad L_\Gamma - T_\Gamma = \infty; \quad R_\Gamma = 0$
while $\Lambda \neq \emptyset$ **do**
  Get v^* first customer in Λ
  $\Lambda = \Lambda \setminus \{v^*\}$
  if $\Gamma = \emptyset$ **then**
    $\Gamma = \{v^*\}; \quad L_\Gamma - T_\Gamma = \kappa L_{v^*} - T_{0v^*}; \quad R_\Gamma = R_{v^*}$
  else
    $L_\Gamma - T_\Gamma = \min\{\kappa L_{v^*} - T_{0v^*}, L_\Gamma - T_\Gamma\}$
    if $R_{v^*} \leq L_\Gamma - T_\Gamma$ **then**
      $\Gamma = \Gamma \cup v^*; \quad R_\Gamma = R_{v^*}$
    else
      $\lambda = \Lambda \cup \{v^*\}$
      for all $v \in \Gamma$ **do**
        $R_v = R_\Gamma$
      end for
        $\Gamma = \emptyset$
    end if
  end if
end while

$$r = 1 - \frac{L_i - R_i^{(2)}}{T_H}. \tag{21}$$

Equation (21) (instead of Equation (1)) allows to univocally determine the value of $R_i^{(2)}$ once r is given. This is done in order to make instance replication easier. Inverting Equation (21) we obtain

$$R_i^{(2)} = L_i + T_H(r-1);$$

from which follows that

$$R_i^{(2)} \geq 0 \Leftrightarrow L_i + T_H(r-1) \geq 0 \Leftrightarrow r \geq 1 - \frac{L_i}{T_H}.$$

Then, when $r < 1 - \frac{L_i}{T_H}$, $R_i^{(2)}$ is negative. We fix the release date to zero in this case. Moreover, the decimal parts are truncated in order to work with integer values. Thus, the following scheme is adopted

$$R_i^{(2)} = \begin{cases} \lfloor L_i + T_H(r-1) \rfloor & \text{if } 1 - \frac{L_i}{T_H} \leq r; \\ 0 & \text{otherwise.} \end{cases}$$

In the second step (lines 8–12 of Algorithm 2), we guarantee that each customer can be served by a round trip, then $R_i^{(1)}$ is calculated as follows

$$R_i^{(1)} = \begin{cases} R_i^{(2)} & \text{if } R_i^{(2)} + T_{0i} \leq L_i, \\ 0 & \text{otherwise.} \end{cases}$$

In the third step (lines 13–32 of Algorithm 2) the final release date values R_i, $i = 1, \ldots, N$ are determined. Customers are *clustered* with respect to the release dates with the purpose to represent different truck arrivals at the depot. All the customers that belong to the same cluster Γ will be associated with the same release date R_Γ.

A list Λ is initialized with all the customers ordered by non-decreasing values of $R^{(1)}$, i.e., such that $i < j$ implies $R_i^{(1)} \leq R_j^{(1)}$. Clusters Γ are constructed starting with a single customer v^1 and successively adding the following customers in Λ. A customer v^2 is added if and only if each customer $v \in \Gamma$ can be served by a round trip even if the merchandise is available at the depot at time $R_{v^2}^{(1)} (\geq R_v^{(1)})$, namely, if and only if

$$R_{v^2}^{(1)} + T_{0v} \leq L_v \quad \forall v \in \Gamma. \tag{22}$$

The final release date R_v of each customer v in Γ is set to $R_{v^2}^{(1)}$. When the next customer v^2 to be inserted in Γ does not satisfy Relation (22), a new cluster is initialized and the procedure restarted.

In order to create different classes of instances, a parameter $\kappa \leq 1$ is introduced in Equation (22). Hence, it becomes

$$R_{v^2}^{(1)} + T_{0v} \leq \kappa L_v \quad \forall v \in \Gamma.$$

Different values of κ produce instances with different rigidity. In particular the higher is κ, the higher is the rigidity r.

The first step release dates $R_i^{(2)}$, $i = 1, \ldots, N$ are determined setting $r = 0.5$. Then, for each Solomon instance, three instances are created using Algorithm 2 with values $\kappa = 0.25, 0.5, 0.75$. A fourth instance has all the release dates equal to zero and will be referred by $\kappa = 0$ for simplicity. There are in total 224 new instances.

Finally, vehicle's capacities are half of the original values, while the number of vehicles M is set in order to have feasible or quasi-feasible solution for the $\kappa = 0.75$ instance group. Since we force some release dates to be zero and we use different values of κ, the final rigidity of instances differs from 0.5. Values of the number of vehicles, actual rigidity r and tightness $tigh$ are reported in Table 2.

		$\kappa = 0$		$\kappa = 0.25$		$\kappa = 0.5$		$\kappa = 0.75$	
	M	$tigh$	r	$tigh$	r	$tigh$	r	$tigh$	r
C101	12	0	0.606	0.085	0.664	0.129	0.690	0.161	0.709
C102	10	0	0.493	0.164	0.629	0.201	0.651	0.225	0.666
C103	11	0	0.350	0.261	0.576	0.290	0.595	0.311	0.609
C104	13	0	0.227	0.350	0.540	0.361	0.548	0.380	0.561
C105	11	0	0.579	0.098	0.647	0.152	0.678	0.190	0.702
C106	11	0	0.563	0.105	0.638	0.152	0.667	0.203	0.697
C107	11	0	0.553	0.102	0.627	0.151	0.656	0.206	0.689
C108	10	0	0.522	0.116	0.607	0.158	0.632	0.225	0.674
C109	10	0	0.467	0.139	0.573	0.192	0.605	0.257	0.644
R101	22	0	0.537	0.093	0.605	0.099	0.609	0.135	0.632
R102	19	0	0.424	0.179	0.571	0.185	0.574	0.222	0.601
R103	18	0	0.332	0.258	0.548	0.261	0.550	0.311	0.589
R104	17	0	0.241	0.332	0.522	0.334	0.523	0.387	0.566
R105	17	0	0.492	0.111	0.575	0.122	0.582	0.181	0.619
R106	15	0	0.391	0.192	0.548	0.201	0.553	0.260	0.594
R107	16	0	0.311	0.266	0.533	0.272	0.536	0.346	0.590
R108	16	0	0.231	0.335	0.514	0.336	0.515	0.382	0.550
R109	15	0	0.427	0.137	0.531	0.147	0.537	0.212	0.577
R110	15	0	0.362	0.204	0.517	0.223	0.527	0.299	0.575
R111	15	0	0.328	0.241	0.521	0.249	0.526	0.320	0.575
R112	18	0	0.283	0.289	0.500	0.314	0.515	0.469	0.618
RC101	19	0	0.492	0.104	0.569	0.120	0.577	0.175	0.611
RC102	17	0	0.406	0.179	0.546	0.187	0.551	0.244	0.587
RC103	18	0	0.332	0.249	0.532	0.257	0.537	0.325	0.584
RC104	19	0	0.256	0.317	0.514	0.317	0.515	0.391	0.570
RC105	18	0	0.430	0.153	0.542	0.167	0.550	0.220	0.584
RC106	16	0	0.427	0.141	0.531	0.156	0.540	0.227	0.584
RC107	18	0	0.365	0.202	0.514	0.212	0.520	0.280	0.563
RC108	18	0	0.302	0.268	0.501	0.280	0.508	0.380	0.571
C201	3	0	0.519	0.151	0.634	0.199	0.664	0.258	0.702
C202	4	0	0.393	0.239	0.600	0.274	0.623	0.318	0.651
C203	5	0	0.277	0.318	0.570	0.339	0.584	0.364	0.601
C204	5	0	0.147	0.405	0.530	0.416	0.538	0.431	0.548
C205	3	0	0.495	0.159	0.619	0.210	0.652	0.262	0.685
C206	3	0	0.469	0.175	0.607	0.225	0.639	0.287	0.679
C207	3	0	0.452	0.185	0.603	0.233	0.634	0.283	0.666
C208	3	0	0.445	0.183	0.592	0.237	0.626	0.290	0.660
R201	4	0	0.493	0.133	0.600	0.162	0.617	0.215	0.651
R202	2	0	0.365	0.222	0.567	0.247	0.581	0.284	0.606
R203	3	0	0.252	0.312	0.546	0.328	0.555	0.355	0.575
R204	3	0	0.141	0.395	0.519	0.397	0.521	0.420	0.540
R205	3	0	0.430	0.177	0.571	0.236	0.607	0.297	0.645
R206	3	0	0.318	0.262	0.548	0.313	0.578	0.365	0.612
R207	3	0	0.222	0.337	0.533	0.368	0.551	0.405	0.578
R208	3	0	0.126	0.409	0.514	0.410	0.514	0.456	0.548
R209	3	0	0.370	0.211	0.537	0.283	0.580	0.380	0.640
R210	3	0	0.334	0.250	0.540	0.310	0.578	0.381	0.623
R211	4	0	0.303	0.276	0.517	0.390	0.582	0.541	0.673
RC201	4	0	0.488	0.142	0.600	0.183	0.624	0.240	0.660
RC202	5	0	0.366	0.232	0.569	0.264	0.588	0.314	0.621
RC203	3	0	0.257	0.320	0.549	0.344	0.563	0.385	0.594
RC204	4	0	0.147	0.393	0.519	0.394	0.519	0.440	0.554
RC205	4	0	0.426	0.176	0.566	0.254	0.610	0.334	0.658
RC206	3	0	0.424	0.172	0.562	0.227	0.594	0.293	0.638
RC207	4	0	0.362	0.212	0.531	0.282	0.571	0.388	0.638
RC208	4	0	0.291	0.276	0.508	0.379	0.568	0.537	0.666

Table 2: Instance details

5.2 Tuning

The procedure makes use of some parameters that need to be set to values chosen into sensible ranges. After conducting preliminary tests we decided to fix the values of ζ^+_{ref} and ζ^-_{ref} to 0.35 and 0.25 respectively, while θ and λ are initially set respectively to 20 and 2.

In order to determine the values of the remaining parameters, we run the Evolutionary Strategy with Covariance Matrix Adaptation proposed by Hansen and Ostermeier [10] on a limited set of instances. In particular $\mathcal{A}^{CAF}$ is run on C108, R104, RC106, RC208, obtained with $\kappa = 0.75$, and we obtained the values reported in Table 3.

Parameter		Range	Final value
π	Dimension of population	$[1, 100]$	20
μ	Children generated at each generation	$[1, 100]$	30
n^e	Proportion of elite individuals $n_e = n^e \times \Pi$ (Eq. 19)	$[0.1, 1]$	0.20
n^c	Proportion of close individuals $n_c = n^c \times \Pi$ (Eq. 20)	$[0.1, 1]$	0.35

Table 3: Parameter Tuning

5.3 Setting of $\mathcal{L}_{\text{threshold}}$

The value of $\mathcal{L}_{\text{threshold}}$ is important to achieve the best compromise between solution quality and computational efficiency. To find a suitable value, we evaluate the impact of $\mathcal{L}_{\text{threshold}}$ on a set of 100 chromosomes. In order to avoid completely random chromosomes, we proceed as follows. A chromosome is randomly generated for instances C101 and C201 of group $\kappa = 0$. *AdSplit* first evaluates them with $\mathcal{L}_{\text{threshold}} = 10$, then they are improved by LS. The resulting chromosomes are re-evaluated by *AdSplit* with different values of $\mathcal{L}_{\text{threshold}}$. Average results obtained on the 100 evaluations are reported in Table 4. The value of

	C101 $\kappa = 0$				C201 $\kappa = 0$			
$\mathcal{L}_{\text{threshold}}$	time (ms)	cost	time gap (%)	cost gap (%)	time (ms)	cost	time gap (%)	cost gap (%)
500	292111	2561.17	-	-	119933	2089.89	-	-
50	5446	2564.93	-98.14	0.15	2797	2089.93	-97.67	≈ 0
45	4299	2565.04	-98.53	0.15	2413	2089.93	-97.99	≈ 0
40	3134	2565.32	-98.93	0.16	2080	2089.93	-98.27	≈ 0
35	2273	2565.42	-99.22	0.17	1698	2089.93	-98.58	≈ 0
30	1550	2565.82	-99.47	0.18	1377	2089.93	-98.85	≈ 0
25	1013	2594.49	-99.65	1.30	1035	2089.93	-99.14	≈ 0
20	631	2656.85	-99.78	3.74	657	2089.93	-99.45	≈ 0
15	346	2713.37	-99.88	5.94	370	2089.93	-99.69	≈ 0
10	164	2788.42	-99.94	8.87	178	2089.93	-99.85	≈ 0
5	62	3185.31	-99.98	24.37	67	2089.93	-99.94	≈ 0
1	14	8619.91	-100.00	236.56	11	2242.80	-99.99	7.32

Table 4: Setting $\mathcal{L}_{\text{threshold}}$

$\mathcal{L}_{\text{threshold}}$ is indicated in the first column of the table. A maximum of 500 labels is considered for $\mathcal{L}_{\text{threshold}}$. In this case, computational times are huge implying also that splitting chromosomes using the Strong Dominance Rule 1 is not time efficient. The Weak Dominance Rule 2 allows a quick evaluation preserving solution quality even with a few labels kept associated with each node. Symbol ≈ 0 means the value is approximately zero. Results show that for instance C201 a small cost deterioration is achieved when the value of $\mathcal{L}_{\text{threshold}}$ is very small. This can be explained by the fact that C201 is characterized by a low number of vehicles, that reduces the possible assignment of trips. Finally, it has been decided to set $\mathcal{L}_{\text{threshold}}= 15$.

5.4 Results on MTVRPTW-R instances

$\mathcal{A}^{CAF}$ is run 5 times over the 224 instances. Each run is stopped after 5 minutes of computation time. Complete results are reported in Tables 5–6, each table being divided in two parts, exhibiting results for instances with $\kappa = 0, 0.25, 0.5, 0.75$. Column *instance* contains the name of the original Solomon instance. Columns *best* report the distance (*dist*) and the number of trips (*#trips*) of the best solution found on the 5 runs. Columns *average* report average distances and trips on the five found solutions. Column *#feas* indicates the number of runs the procedure found a feasible solution. A dash means no feasible solution has been found for the respective instance. It can be noticed that the best solution can be formed by a number of trips higher than the average.

Result analysis is reported on Table 7 and in Figure 3. Table 7 reports average results per group of instances. Columns *best* report average distance and number of trips of the corresponding best solution found by the procedure, while columns *average* report the average of the average values on instances of the same group. Column *% feas* indicates the percentage of feasible solutions found on the total number of runs (that is 5 times the instances forming a specified group).

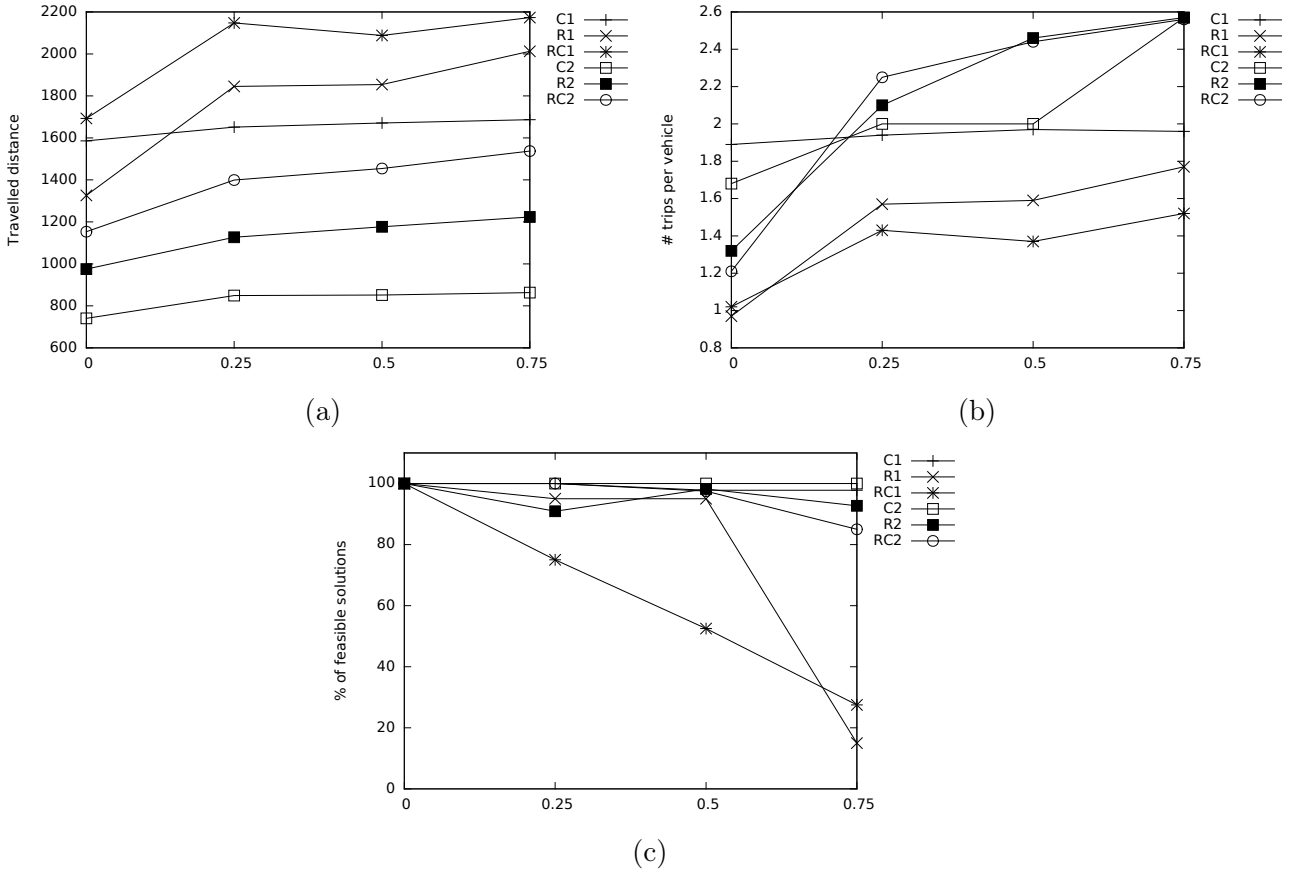


Figure 3: Result analysis

Procedure $\mathcal{A}^{CAF}$ always finds a feasible solution for instances with $\kappa = 0$ (Table 5 and 7). This validates the generation scheme presented in Section 5.1.

Rigidity and tightness of instances *averagely* describe the difficulty of solving an instance: considering results grouped by type of instances and value of κ, it can be noticed (Figures 3a–3c, Table 7) that when the tightness and rigidity grow, the travelled distance and the number of trips per vehicle grow, while the number of feasible solutions found by the algorithm decreases. On the other side, there is no evident punctual correlation between rigidity and tightness, and the results obtained on a specific instance: for example instance R103 with $\kappa = 0.25$ has higher tightness and rigidity than instance RC102 (see Table 2),

instance	$\kappa = 0$					$\kappa = 0.25$				
	best		average		#feas	best		average		#feas
	dist	#trips	dist	#trips		dist	#trips	dist	#trips	
C101	1529.66	20	1568.27	20.8	5	1565.24	21	1585.29	21.0	5
C102	1675.75	22	1715.14	21.8	5	1759.23	22	1778.12	22.4	5
C103	1452.69	19	1524.98	19.6	5	1748.96	22	1810.23	22.6	5
C104	1384.78	19	1407.10	19.2	5	1774.75	22	1834.28	22.6	5
C105	1550.02	21	1709.18	22.2	5	1552.38	21	1601.48	21.4	5
C106	1592.13	20	1624.38	21.4	5	1594.68	21	1621.70	21.6	5
C107	1513.19	20	1527.41	20.0	5	1511.36	20	1536.45	20.0	5
C108	1545.94	21	1659.66	21.6	5	1518.79	20	1548.04	20.2	5
C109	1496.65	21	1538.70	20.2	5	1536.90	21	1547.45	20.4	5
R101	1671.77	22	1678.46	20.6	5	2102.53	32	2160.65	31.4	5
R102	1498.23	18	1502.58	18.0	5	2133.22	31	2146.89	32.5	2
R103	1288.44	16	1298.49	16.0	5	1924.76	27	1990.82	29.0	5
R104	1177.88	15	1190.59	15.0	5	1632.43	24	1711.13	25.6	5
R105	1421.19	16	1433.05	16.4	5	1848.76	25	1913.50	26.6	5
R106	1361.02	16	1365.95	16.2	5	1882.07	27	1959.79	28.4	5
R107	1235.15	16	1245.09	16.0	5	1830.08	26	1861.37	27.2	5
R108	1187.36	15	1193.28	15.0	5	1565.98	22	1622.82	23.0	5
R109	1307.25	17	1316.19	16.4	5	1750.73	25	1835.34	26.0	5
R110	1246.99	15	1253.13	15.0	5	1741.20	25	1779.37	26.6	5
R111	1236.23	17	1244.99	16.2	5	1803.39	26	1833.50	26.4	5
R112	1182.72	15	1191.47	15.6	5	1323.48	17	1329.28	17.0	5
RC101	1805.40	19	1828.30	19.0	5	2304.70	26	2398.29	28.6	5
RC102	1746.02	18	1759.63	18.2	5	-	-	-	-	0
RC103	1637.38	18	1641.92	18.0	5	2161.58	25	2319.11	28.0	5
RC104	1582.81	18	1583.46	18.0	5	1884.44	22	1963.69	22.8	5
RC105	1752.66	19	1759.14	18.4	5	2291.55	28	2367.93	29.0	2
RC106	1750.52	19	1764.37	18.8	5	2249.39	29	2266.68	28.0	3
RC107	1615.05	18	1618.37	18.0	5	1911.32	21	1980.79	22.8	5
RC108	1581.78	18	1587.03	18.0	5	1706.06	20	1737.62	19.4	5
C201	777.48	6	777.48	6.0	5	781.76	7	781.76	7.0	5
C202	718.69	6	724.85	6.0	5	913.97	7	914.23	7.0	5
C203	700.20	6	711.06	6.0	5	949.71	8	949.71	8.0	5
C204	695.12	6	698.17	6.0	5	966.98	7	977.18	7.2	5
C205	767.55	7	770.21	6.8	5	755.45	7	755.45	7.0	5
C206	747.14	6	750.42	6.0	5	796.57	7	797.31	7.0	5
C207	746.62	6	748.66	6.0	5	786.64	7	788.08	7.0	5
C208	741.58	6	742.09	6.0	5	820.57	8	828.71	7.8	5
R201	1272.47	4	1287.21	4.2	5	1403.33	8	1444.78	9.0	5
R202	1272.72	4	1278.20	4.2	5	1400.45	6	1452.05	6.6	5
R203	966.35	5	976.30	4.2	5	1140.24	6	1162.86	6.0	5
R204	779.22	3	787.31	3.6	5	1018.57	5	1027.23	5.8	5
R205	1074.75	5	1089.24	4.6	5	1141.30	8	1163.15	7.2	5
R206	944.58	4	962.93	4.2	5	1018.97	6	1034.26	6.0	5
R207	849.64	4	862.07	4.0	5	981.61	6	993.27	6.0	5
R208	735.49	4	738.52	3.8	5	905.53	4	912.03	4.6	5
R209	944.06	3	961.83	3.6	5	1050.79	6	1127.31	6.8	5
R210	985.66	4	1001.98	4.2	5	1149.92	7	1178.50	7.0	5
R211	772.99	5	779.80	4.2	5	891.09	6	900.95	6.6	5
RC201	1424.18	5	1459.26	5.2	5	1637.82	9	1690.93	9.8	5
RC202	1171.86	4	1196.66	4.4	5	1423.40	8	1508.23	9.4	5
RC203	1108.21	4	1150.94	5.0	5	1441.46	6	1485.27	8.2	5
RC204	806.44	4	809.59	4.2	5	1076.24	7	1082.30	6.8	5
RC205	1321.64	4	1354.18	4.6	5	1543.24	11	1585.42	10.4	5
RC206	1325.01	5	1385.87	5.4	5	1493.13	9	1554.81	9.8	5
RC207	1042.03	4	1050.24	4.6	5	1185.78	8	1228.64	8.2	5
RC208	803.59	4	818.23	4.0	5	1001.06	6	1060.34	7.0	5

Table 5: Results on new instances, $\kappa = 0$ and $\kappa = 0.25$

instance	$\kappa = 0.5$					$\kappa = 0.75$				
	best		average		#feas	best		average		#feas
	dist	#trips	dist	#trips		dist	#trips	dist	#trips	
C101	1579.19	22	1584.99	21.8	5	1591.91	22	1611.34	22.2	5
C102	1746.59	22	1774.92	22.4	5	1766.15	22	1793.42	22.2	5
C103	1842.84	24	1889.73	23.6	5	1900.10	23	1920.27	24.5	4
C104	1773.04	21	1814.93	21.8	5	1805.55	20	1890.63	21.8	5
C105	1603.96	22	1637.48	22.0	4	1600.94	21	1633.32	21.4	5
C106	1587.25	22	1605.18	21.8	5	1663.38	22	1695.00	21.8	5
C107	1518.75	20	1529.53	20.0	5	1538.41	20	1547.13	20.0	5
C108	1556.50	20	1662.16	21.4	5	1546.53	20	1569.45	20.2	5
C109	1506.85	20	1541.79	20.0	5	1512.26	20	1521.20	20.2	5
R101	2135.93	32	2236.49	33.0	5	2372.83	35	2372.83	35.0	1
R102	2102.70	31	2120.94	32.0	2	-	-	-	-	0
R103	1879.98	27	1959.64	28.6	5	-	-	-	-	0
R104	1637.50	22	1680.34	23.8	5	-	-	-	-	0
R105	1964.79	27	1987.37	28.6	5	2046.53	31	2135.91	32.0	2
R106	1900.14	28	1964.97	28.6	5	2043.12	30	2043.12	30.0	1
R107	1853.74	27	1888.95	27.6	5	-	-	-	-	0
R108	1562.17	22	1618.08	23.2	5	-	-	-	-	0
R109	1730.47	24	1819.57	26.0	5	1898.65	28	1898.65	28.0	1
R110	1682.08	24	1756.21	25.8	5	-	-	-	-	0
R111	1784.27	26	1884.33	27.6	5	-	-	-	-	0
R112	1320.07	17	1332.57	17.4	5	1540.39	23	1612.44	24.8	4
RC101	2484.09	30	2484.09	30.0	1	-	-	-	-	0
RC102	-	-	-	-	0	-	-	-	-	0
RC103	2194.26	25	2270.77	26.8	5	-	-	-	-	0
RC104	1896.29	22	1930.99	22.6	5	2175.03	27	2214.98	27.3	4
RC105	-	-	-	-	0	-	-	-	-	0
RC106	-	-	-	-	0	-	-	-	-	0
RC107	1918.73	21	2024.90	24.0	5	2249.07	28	2290.20	28.6	5
RC108	1718.04	19	1728.73	19.4	5	1985.12	24	2014.37	25.5	2
C201	788.37	7	788.37	7.0	5	815.58	6	815.58	6.0	5
C202	913.66	7	914.14	7.0	5	913.66	7	915.58	7.0	5
C203	952.09	8	962.94	8.0	5	952.46	8	952.47	8.0	5
C204	967.23	7	975.28	7.0	5	976.79	7	982.89	7.0	5
C205	762.06	7	762.06	7.0	5	778.45	6	778.45	6.0	5
C206	796.57	7	797.32	7.0	5	813.52	6	813.52	6.0	5
C207	784.22	7	789.49	7.0	5	805.76	6	806.23	6.2	5
C208	817.35	8	824.63	8.0	5	833.46	8	841.61	7.6	5
R201	1443.84	10	1464.64	8.6	5	1430.19	9	1455.41	8.6	5
R202	1425.40	9	1452.19	8.3	4	1452.75	9	1481.26	8.3	3
R203	1214.24	7	1242.41	7.6	5	1255.53	8	1287.27	8.6	5
R204	990.54	6	1022.97	5.8	5	987.98	6	1028.77	6.0	5
R205	1183.48	8	1256.04	9.2	5	1242.05	9	1266.00	9.0	5
R206	1069.98	7	1113.78	8.2	5	1111.86	8	1160.79	8.2	5
R207	1004.76	6	1032.92	6.8	5	1034.82	7	1054.83	6.8	5
R208	905.90	4	920.82	4.6	5	910.47	5	944.81	5.6	5
R209	1188.91	9	1237.35	9.6	5	1320.07	9	1327.13	9.3	3
R210	1228.73	9	1288.66	8.4	5	1268.23	8	1338.17	9.2	5
R211	902.45	6	910.85	6.6	5	1074.15	8	1114.96	7.8	5
RC201	1677.63	10	1784.00	11.8	5	1796.39	10	1874.02	11.4	5
RC202	1427.31	10	1522.27	10.6	5	1539.80	13	1599.26	12.2	5
RC203	1464.58	8	1480.79	8.8	4	1488.91	9	1493.40	9.0	2
RC204	1084.93	7	1086.09	7.0	5	1103.65	7	1123.24	7.2	5
RC205	1695.25	11	1734.99	11.4	5	1777.28	12	1808.49	11.4	5
RC206	1445.43	9	1575.79	9.0	5	1493.88	9	1594.45	9.2	5
RC207	1163.13	8	1328.49	9.6	5	1449.89	11	1502.53	10.6	5
RC208	1107.69	8	1122.44	7.6	5	1297.58	9	1304.99	8.5	2

Table 6: Results on new instances, $\kappa = 0.5$ and $\kappa = 0.75$

Instance Group	best		average		% feas	tight	rigidity
	dist	#trips	dist	#trips			
	$\kappa = 0$						
C1	1526.76	20.33	1586.09	20.76	100	0.00	0.48
R1	1317.85	16.50	1326.11	16.37	100	0.00	0.36
RC1	1683.95	18.38	1692.78	18.30	100	0.00	0.38
C2	736.80	6.13	740.37	6.10	100	0.00	0.40
R2	963.45	4.09	975.04	4.07	100	0.00	0.30
RC2	1125.37	4.25	1153.12	4.68	100	0.00	0.35
	$\kappa = 0.25$						
C1	1618.03	21.11	1651.45	21.36	100	0.16	0.61
R1	1794.89	25.58	1845.37	26.64	95.0	0.22	0.54
RC1	2072.72	24.43	2147.73	25.51	75.0	0.20	0.53
C2	846.45	7.25	849.05	7.25	100	0.23	0.59
R2	1100.16	6.18	1126.94	6.51	100	0.27	0.54
RC2	1350.27	8.00	1399.49	8.70	100	0.21	0.53
	$\kappa = 0.5$						
C1	1635.00	21.44	1671.19	21.64	97.8	0.20	0.64
R1	1796.15	25.58	1854.12	26.85	95.0	0.23	0.55
RC1	2042.28	23.40	2087.90	24.56	52.5	0.21	0.54
C2	847.69	7.25	851.78	7.25	100	0.27	0.62
R2	1141.66	7.36	1176.60	7.60	98.2	0.31	0.57
RC2	1383.24	8.88	1454.36	9.47	97.5	0.28	0.58
	$\kappa = 0.75$						
C1	1658.36	21.11	1686.86	21.59	97.8	0.24	0.66
R1	1980.30	29.40	2012.59	29.95	15.0	0.29	0.59
RC1	2136.41	26.33	2173.18	27.12	27.5	0.28	0.58
C2	861.21	6.75	863.29	6.73	100	0.31	0.65
R2	1189.83	7.82	1223.58	7.95	92.7	0.37	0.61
RC2	1493.42	10.00	1537.55	9.94	85.0	0.34	0.62

Table 7: Statistics on new instances

but 5 feasible solutions out of 5 runs are found for the former, while none for the latter.

Robustness of procedure $\mathcal{A}^{CAF}$ is proved by the small differences between best values and average values reported in Tables 5–6 and in Table 7.

5.5 Comparison with Hernandez et al. [12]

To evaluate the performance of $\mathcal{A}^{CAF}$ we run the procedure on instances generated by Hernandez et al. [12] for the MTVRP with TW. These instances are generated from Solomons's instances in groups C2, R2, RC2, considering the first 25 customers and M fixed to 2, and the first 50 customers and $M = 4$. Vehicle capacity is fixed to 100, loading time at the depot is trip dependent and in particular it is 0.2 times the sum of service times at customers in the trip. Travel times are the Euclidean distances rounded to the first decimal. Limitation into the number of customers is due to the exact nature of the algorithm proposed by Hernandez et al. [12]. Due to the heuristic nature of our algorithm, we consider as well the instances with all the 100 customers. Following the instance generation system of Hernandez et al. [12], we double the number of available vehicles used for instances with 50 customers. Then, 8 vehicles are available to serve the 100 customers.

Instances in groups C1, R1 and RC1 are not considered by Hernandez et al. [12] due to short time horizon that, in their opinion, would not allow vehicles to perform different trips.

$\mathcal{A}^{CAF}$ is run five times on each instance and it is stopped after 1 minute on instances with 25 customers and after 5 minutes on instances with 50 customers and with 100 customers. Results are reported in Tables 8–10.

The first column reports the instance name, columns *HRN* report the optimal value (column *opt*) found by Hernandez et al. [12]. A blank indicates they could not find the optimal solution. In some cases their algorithm provides a feasible solution which value is indicated in column *feas*.

Columns *best* report the travelled distance (*dist*) and the number of trips (*# trips*) that characterize the best solution found by $\mathcal{A}^{CAF}$in the five runs. Columns *average* indicate average values over the five runs. Bold numbers indicate the best known solution has been improved by $\mathcal{A}^{CAF}$ (we omitted the bold font when no solution value was available). Finally, column *# opt* reports the number of runs $\mathcal{A}^{CAF}$ finds the optimal solution on the five runs. A dash is reported when the optimal value is not available.

It can be observed that on small instances $\mathcal{A}^{CAF}$ finds the optimal solution on all the five runs on 23 out of 25 instances. It fails in finding the optimal on only 2 runs in total, one for instance C201 and one for instance RC206. The average gap from the optimal value is respectively 0.079% and 0.003%. Moreover, a new best solution is obtained for instance RC204. On instances with 50 customers, our procedure fails in finding the optimal solution only for instance RC202, while in the other four cases the optimal solution is retrieved 14 times out of 20 runs. 8 instances on the 9 with a feasible known solution are improved, while in the remaining case a same cost solution is got. The average percentage gap from the optimal value is 0.03%. Feasible solutions are found for all the instances, included those with the all 100 customers.

6 Conclusions and perspectives

In this paper we introduced a new problem, the Multi Trip Vehicle Routing Problem with Time Windows and Release Dates. It raises in city logistics context, where trucks deliver goods to city distribution centers (CDC) before they are delivered to final customers by eco-friendly vans. Optimization of van trips depends on the truck delivery plan to the CDC. Trucks arrive during the whole day, continuously bringing goods into the distribution system. Arrival of trucks to CDC is modelled associating a *release date* with each merchandise. It represents the moment the merchandise itself becomes available for final delivery.

We introduced a new set of instances on which we run the memetic algorithm we developed. Moreover, we run the algorithm on instances for the Multi Trip Vehicle Routing Problem with Time Windows for

Instance	HRN		best		average		#opt
	opt	feas	dist	# trips	dist	# trips	
C201	380.8		380.8	3	381.10	5.4	4
C202	368.6		368.6	5	368.60	5.0	5
C203	361.7		361.7	5	361.70	5.0	5
C204	358.8		358.8	5	358.80	5.0	5
C205	377.2		377.2	5	377.20	5.0	5
C206	367.2		367.2	5	367.20	5.0	5
C207	359.1		359.1	5	359.10	5.0	5
C208	360.9		360.9	5	360.90	5.0	5
R201	554.6		554.6	4	554.60	4.0	5
R202	485.0		485.0	4	485.00	4.0	5
R203	444.2		444.2	4	444.20	4.0	5
R204	407.5		407.5	4	407.50	4.0	5
R205	448.4		448.4	4	448.40	4.0	5
R206	413.9		413.9	4	413.90	4.0	5
R207	400.1		400.1	4	400.10	4.0	5
R208	394.3		394.3	4	394.30	4.0	5
R209	418.3		418.3	4	418.30	4.0	5
R210	448.3		448.3	4	448.30	4.0	5
R211	400.1		400.1	4	400.10	4.0	5
RC201	660.0		660.0	6	660.00	6.0	5
RC202	596.8		596.8	6	596.80	6.0	5
RC203	530.1		530.1	6	530.10	6.0	5
RC204		520.3	**518.0**	6	518.00	6.0	-
RC205	605.3		605.3	6	605.30	6.0	5
RC206	575.1		575.1	6	575.12	6.0	4
RC207	528.2		528.2	6	528.20	6.0	5
RC208		506.4	506.4	6	506.40	6.0	-

Table 8: Results on Hernandez et al. [12] instances with $N = 25$ and $M = 2$

Instance	HRN		best		average		#opt
	opt	feas	dist	# trips	dist	# trips	
C201		717.9	**714.2**	10	714.20	10.0	-
C202		701.9	**700.1**	9	700.38	9.0	-
C203			688.0	9	689.34	9.0	-
C204			685.1	9	685.10	9.0	-
C205		706.6	**700.0**	9	703.52	9.8	-
C206			694.6	9	696.92	9.2	-
C207			689.7	9	690.38	9.0	-
C208			688.6	9	688.60	9.0	-
R201	909.8		909.8	9	917.08	9.0	1
R202	816.0		816.0	8	816.00	8.0	5
R203			742.4	8	743.40	8.0	-
R204			702.3	8	704.38	8.0	-
R205	807.3		807.3	8	808.74	8.0	3
R206		767.6	**758.2**	8	760.96	8.0	-
R207			715.7	8	715.70	8.0	-
R208			699.6	8	700.60	8.0	-
R209		749.6	**746.0**	8	746.00	8.0	-
R210			777.2	8	779.22	8.0	-
R211			717.4	8	722.02	8.0	-
RC201	1096.6		1096.6	10	1096.60	10.0	5
RC202	1001.6		1038.6	10	1038.60	10.0	0
RC203		945.8	**941.2**	10	941.20	10.0	-
RC204		915.9	915.9	10	915.90	10.0	-
RC205		1065.4	**1058.7**	10	1058.70	10.0	-
RC206			1027.4	11	1032.12	10.8	-
RC207		944.8	**941.7**	10	941.70	10.0	-
RC208			916.8	10	916.80	10.0	-

Table 9: Results on Hernandez et al. [12] instances with $N = 50$ and $M = 4$

Instance	best		average	
	dist	# trips	dist	# trips
C201	1488.9	19	1500.22	19.2
C202	1479.3	19	1486.94	19.0
C203	1467.3	19	1471.20	19.0
C204	1453.6	19	1455.46	19.0
C205	1477.1	19	1483.04	19.0
C206	1464.7	19	1473.38	19.0
C207	1464.2	19	1470.36	19.0
C208	1459.4	19	1465.86	19.0
R201	1449.7	16	1464.32	15.8
R202	1343.3	16	1352.70	15.8
R203	1222.2	15	1232.50	15.4
R204	1165.6	15	1172.54	15.0
R205	1292.2	15	1315.08	15.6
R206	1239.9	15	1249.76	15.4
R207	1194.3	15	1200.76	15.0
R208	1159.8	15	1164.56	15.0
R209	1234.5	16	1248.42	15.4
R210	1247.5	15	1253.24	15.6
R211	1170.5	15	1182.38	15.0
RC201	1843.6	18	1862.40	18.8
RC202	1733.9	18	1740.34	18.2
RC203	1618.6	18	1624.24	18.2
RC204	1579.1	18	1581.36	18.0
RC205	1759.8	18	1776.68	18.0
RC206	1731.1	18	1747.24	18.0
RC207	1656.7	18	1662.96	18.0
RC208	1580.9	18	1585.44	18.0

Table 10: Results on instances with $N = 100$ and $M = 8$ created as in Hernandez et al. [12]

performance evaluation purposes. Results show the efficiency of our procedure.

An efficient labelling procedure is proposed to turn permutation of customers into solution that is an adaptation of the procedure proposed by Prins [26] for the VRP. It is designed for the MTVRPTW-R case, but it can be used in the MTVRPTW context as well.

Associating a release date with each merchandise implicitly suppose the arrival of each truck to the depot is known in advance, at least before the operational planning is computed. Communication and organization between carriers and the management center is needed. Future studies could introduce some dynamism in the problem, considering part of the goods or the whole merchandise to arrive at the depot with no advanced notice. Optimization procedure needs to react to these events, reorganizing the planning quickly and efficiently.

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] 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.

[2] N. Azi, M. Gendreau, and J.-Y. Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 202(3):756–763, 2010.

[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. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Węglarz. *Handbook on Scheduling - From Theory to Application*, chapter Scheduling on One Processor. 2007.

[5] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[6] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[7] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, `http://dx.doi.org/10.1016/j.ejor.2013.06.012`, 2013.

[8] T.G. Crainic. *City Logistics*, pages 181–212. Tutorials in Operations Research 2008 - State-of-the-Art Decision Making Tools in the Information-Intensive Age. INFORMS, 2008.

[9] T. Grünert. GTS Systems & Consulting GmbH, 2013. Private communication.

[10] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[11] V.C. Hemmelmayr, J.F. Cordeau, and T.G. Crainic. An adaptive large neighbourhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

[12] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. An exact algorithm to solve the multi-trip vehicle routing problem with time windows. *xxx*, xxx:xxx–xxx, 2013.

[13] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4OR*, `10.1007/s10288-013-0238-z`, 2013.

[14] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[15] S. Irnich. A unified modeling and solution framework for vehicle routing and local search-based metaheuristics. *INFORMS Journal on Computing*, 20(2):270–287, 2008.

[16] A. Kleff. PTV Group, Karlsruhe, Germany, 2013. Private communication.

[17] A. Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Technical University of Denmark (DTU), 2001.

[18] A. Larsen, O.B.G. Madsen, and M.M Solomon. Recent developments in dynamic vehicle routing systems. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, number 43 in Operations Research Computer Science Interfaces, pages 199–218. Springer, 2008.

[19] R. Macedo, C. Alves, J.M. Valério deCarvalho, F. Clautiaux, and S. Hanafi. Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. *European Journal of Operational Research*, 214(3):536–545, 2011.

[20] 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.

[21] Y. Nagata, O. Bräysy, and W. Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724–737, 2010.

[22] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[23] A. Olivera and O. Viera. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, 34(1):28–47, 2007.

[24] V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[25] M.L. Pinedo. *Planning and Scheduling in Manufacturing and Services*, chapter Manufacturing Models, pages 19–36. Springer Series in Operations Research. Springer, 2001.

[26] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[27] C Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6):916–928, 2009.

[28] 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.

[29] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4, 1985.

[30] M. Schneider, B. Sand, and A. Stenger. A note on the time travel approach for handling time windows in vehicle routing problems. *Computers & Operations Research*, 2013.

[31] M.M. Solomon. Algorithms for the vehicle routing and scheduling problem with time windows constraints. *Operations Research*, 35:254–265, 1987.

[32] É.D. Taillard, G. Laporte, and M. Gendreau. Vehicle routeing with multiple use of vehicles. *Journal of Operational Research Society*, 47(8):1065–1070, 1996.

[33] T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[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.