Open access • Journal Article • DOI:10.1287/TRSC.2015.0608

The multi-trip vehicle routing problem with time windows and release dates

Diego Cattaruzza, Nabil Absi, Dominique Feillet

Published on: 21 Jan 2016 - Transportation Science (INFORMS)

Topics: Vehicle routing problem, Static routing, Policy-based routing, Destination-Sequenced Distance Vector routing and Dynamic Source Routing

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

Diego Cattaruzza, Nabil Absi, Dominique Feillet, Olivier Guyon, Xavier Libeaut

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

Diego Cattaruzza[1], Nabil Absi[1], Dominique Feillet[1], Olivier Guyon[1], Xavier Libeaut[2]

[1] Ecole des Mines de Saint-Etienne
CMP Georges Charpak, F. 13541 Gardanne, France
{cattaruzza, absi, feillet, guyon}@emse.fr

[2] LUNAM Université, Université Catholique de l'Ouest, LISA
3 Place André Leroy, 49008 Angers, France
xavier.libeaut@uco.fr

Abstract

In this paper the Multi Trip Vehicle Routing Problem with Time Windows and Release Dates is introduced. The problem is particularly interesting in the City Logistics context, where trucks deliver merchandise to depots located in the outskirts of the city. Goods continuously arrive during the day becoming available for final distribution after the working day has started. This introduces the concept of release dates associated with merchandise. In this paper, a set of instances is introduced and a hybrid genetic algorithm is proposed to solve the problem.

1 Introduction

The well known Vehicle Routing Problem (VRP) is an 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.

In some city distribution systems, goods are delivered to city distribution centers (CDC) and then delivered to final customers by vehicles with limited capacity. They can be back to the depot much earlier than the end of the working day and then be available for another delivery trip. 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 losing reliability and trustworthiness and often penalties need to be payed. 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 completely available at the CDC at the beginning of the planning horizon. The concept of *release date* is then associated with each merchandise.

In this paper we address the Multi Trip Vehicle Routing Problem with Time Windows and Release Dates (MTVRPTWR). It is noteworthy that the problem is *deterministic* even if the merchandise continuously arrives to the CDC during the day: the release dates are supposed to be known before the working day starts.

The paper is structured as follows. Section 2 reviews the literature, Section 3 formally defines the problem. In Section 4 the hybrid genetic algorithm (HGA) we proposed to solve the MTVRPTWR is explained. Results are reported in Section 5, while Section 6 concludes the paper.

2 Literature Review

In multi-echelon distribution systems, delivery to customer is not direct, but goods are firstly delivered to CDC. Final distribution trips need to take into account the moment goods become available at the CDC. Normally, only the availability of the vehicles to perform deliveries is required. When goods must be loaded immediately after unloading, synchronization aspects can be introduced (Drexl [6]).

We model time-interdependency among vehicles introducing the concept of *release date* associated with merchandise. To the best of our knowledge no previous work is done on it.

Few papers study the Multi Trip VRP with Time Windows (MTVRPTW) and most of all use exact methods to solve it.

Azi et al. [1] propose an exact algorithm to solve 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 route and two nodes are connected by an arc whether it is possible to serve the two routes consequently and they do not serve common customers. Solomon 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 with a column generation approach embedded within a branch-and-price algorithm. 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 instances are considered and a time horizon is introduced. Due to the limitation 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. 24 instances out of 27 of size 25 are solved within 30 hours. When the number of customers increases to 40, only 9 instances out of 27 can be solved to optimality. Hernandez et al. [8] use a similar approach of Azi et al. [2]. A set covering formulation is given for the problem and each column represents a trip instead of a working day. Results on the same instances of Azi et al. [2] show that 25 out of 27 instances with 25 customers and large time horizon and 2 vehicles and 22 out of 27 instances of size 50 with large time horizon and 4 vehicles have been solved within 30 hours. All the three exact methods cited use the algorithm proposed by Feillet et al. [7] for the elementary shortest path problem.

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. The approach they propose is a two-stage sequential heuristic: they generate a set of feasible routes considering each commodity independently, then, routes are assigned to vehicles.

3 Problem definition and notation

The MTVRPTWR can be defined on a completed 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 D_i and a Time Window (TW) $[E_i, L_i]$ during which the service should start and a service time S_i. Arriving at customer location before E_i is allowed. Since the service cannot start earlier than E_i the driver must wait. Late arrivals at customer location are forbidden. Moreover, the quantity D_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 say R_i *is associated with customer* i, instead of R_i *is associated with the demand* D_i *requested by customer* i. It is possible to travel from i to j incurring in a travel time T_{ij}. A vehicle, before performing each trip, needs to wait until goods it is going to deliver are available at the depot, i.e., it cannot start the trip before the maximal release date associated with customers it has to serve.

A time horizon T_H is given and establishes the duration of the working day. It must be respected and it can be viewed as a TW associated with the depot. Thus, it is assumed that $[E_0, L_0] = [0, T_H]$. Moreover $D_0 = 0$.

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

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

(b) each route starts not earlier than the largest R_i associated with customers assigned to the route itself;

(c) each customer is visited by exactly one route;

(d) service at customer i starts during the associated TW $[E_i, L_i]$;

(e) the sum of the demands of the customers in any route does not exceed Q;

(f) each vehicle arrives at the depot after performing its last route not later then T_H (TW associated with the depot is respected).

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

The following notation is used in the remaining. The symbol σ indicates a route. The capital Σ indicates the set of routes assigned to a vehicle. For easiness, it is called *journey*. Then, a journey is formed by different routes. The symbol $\oplus$ indicates the concatenation of paths (partial routes) or routes. For example $(v_1, \ldots, v_n) \oplus (w_1, \ldots, w_m)$ is the concatenation of two paths (that results in a route if v_1 and w_m are the depot). $\sigma_1 \oplus \sigma_2$ means that route σ_1 is performed right before σ_2 on the same vehicle. The time a vehicle needs to visit customers on route σ_i is indicated by τ^{σ_i}, while the time the vehicle leaves the depot to perform σ_i is indicated by $\mathcal{T}^{\sigma_i}$.

The symbol "$\in$" describes "belonging". For example $\sigma \in \Sigma$ means route σ belongs to journey Σ, $v \in \sigma$ means customer v belongs to (i.e., it is served by) route σ, and so on.

It is noteworthy that in absence of release dates it would hold $\mathcal{T}^{\sigma_{i+1}} = \mathcal{T}^{\sigma_i} + \tau^{\sigma_i}$. On the other hand, considering release dates it holds $\mathcal{T}^{\sigma_{i+1}} = \max\{\mathcal{T}^{\sigma_i} + \tau^{\sigma_i}, \max_{v_i \in \sigma_i}\{R_{v_i}\}\}$.

A customer v_i is *feasible* if the vehicle arrives at its location before L_i, *infeasible* otherwise. A route is feasible if it serves only feasible customers, infeasible otherwise. A journey is feasible if it is composed by feasible routes, infeasible otherwise.

4 Method

We propose a genetic algorithm (HGA) for the MTVRPTWR. The main scheme of our approach is illustrated in Algorithm 1. The algorithm is an adaptation of the procedure proposed by Cattaruzza et al. [4]

Algorithm 1 Hybrid Gentic Algorithm sketch

```
1:  Initialize population
2:  while Termination criteria are not met do
3:      Select parent chromosomes Ψ_{P_1} and Ψ_{P_2}
4:      Generate a child Ψ_C
5:      Improve Ψ_C with LS
6:      if Ψ_C is infeasible then
7:          Repair Ψ_C
8:      end if
9:      Insert Ψ_C in the population
10:     if Dimension of the population is bigger or equal than π + μ then
11:         Select survivors
12:     end if
13: end while
```

(we refer to this paper for more details). First, π random chromosomes are generated in order to create an initial population. These individuals are then *improved* with Local Search (LS). The classical binary tournament procedure is used for selection and children are created by crossing the selected parents (with the OX operator). Infeasible chromosomes are *repaired* (Section 4.4). When the population reached the dimension of $\pi + \mu$, μ individuals are eliminated accordingly to their quality and their diversification contribution to the population (as proposed by Vidal et al. [13]).

In the remaining, Sections 4.2 and 4.3 provide details for the LS while Section 4.5 describes *AdSplit* procedure used to obtain MTVRPTWR solutions from a chromosome.

4.1 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 in a feasible MTVRPTWR solution by splitting the chromosome (inserting trip delimiters and assigning trips to vehicles). From that point of view, Ψ is usually called a *giant tour*. From a giant tour Ψ, different MTVRPTWR 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 respectively by coefficients λ and θ.

The procedure *AdSplit* (see Section 4.5) is used to get a MTVRPTWR solution ξ from Ψ. The two following notations are introduced: $T_\Sigma(\xi)$ and $TW_\Sigma(\xi)$ are respectively the travel time and the TW violation of journey Σ in solution ξ. $L_\sigma(\xi)$ is the load of trip σ where $\sigma \in \Sigma$ is a route σ of journey Σ. The fitness $F(\Psi)$ of the chromosome Ψ is the cost of the best solution ξ found by *AdSplit* and it is defined as follows:

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

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.

4.2 Local Search for VRP with Time Windows

LS in presence of TW becomes more complicated than in the classic VRP. In particular, feasibility cannot be checked in constant time straightforwardly. In our approach, we adapt the scheme proposed by Vidal et al. [12] (that is in turn an extension of the scheme proposed by Nagata et al. [10]) for a large class of problems with TWs. Given a path ρ, the quantities $T(\rho)$, $TW(\rho)$, $E(\rho)$, $L(\rho)$, $D(\rho)$ respectively denote the minimum duration, the minimum TW violation of ρ, 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 penalization, and the cumulative demand of served customer. Given two paths ρ_1 and ρ_2, the following relations hold:

$$T(\rho_1 \oplus \rho_2) = T(\rho_1) + T(\rho_2) + T_{v_1,v_2} + \Delta_{WT}; \tag{2}$$
$$TW(\rho_1 \oplus \rho_2) = TW(\rho_1) + TW(\rho_2) + \Delta_{TW}; \tag{3}$$
$$E(\rho_1 \oplus \rho_2) = \max\{E(\rho_2) - \Delta, E(\rho_1)\} - \Delta_{WT}; \tag{4}$$
$$L(\rho_1 \oplus \rho_2) = \min\{L(\rho_2) - \Delta, L(\rho_1)\} + \Delta_{TW}; \tag{5}$$
$$D(\rho_1 \oplus \rho_2) = D(\rho_1) + D(\rho_2); \tag{6}$$

where

$$\Delta = T(\rho_1) - TW(\rho_1) + T_{v_1,v_2};$$
$$\Delta_{WT} = \max\{E(\rho_2) - \Delta - L(\rho_1), 0\};$$
$$\Delta_{TW} = \max\{E(\rho_1) + \Delta - L(\rho_2), 0\};$$
$$v_1 \text{ last customer in } \rho_1 \text{ and } v_2 \text{ first customer in } \rho_2.$$

Knowing those quantities allows to evaluate classical LS operators in constant time.

The quantities defined above for all the paths and their reverse (needed, for example, to evaluate 2-opt moves) can be calculated in a preprocessing phase.

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

The problem we address has two more characteristics than the VRPTW: vehicles can perform several trips and goods become available for final distribution throughout the time horizon. The consequence is

that vehicles can start a trip at $t > 0$, because the trip they are going to accomplish is not the first, or either because they wait for goods to be available.

Vidal et al. [12] prove relations 2–6 by induction on the concatenation operator proving the following stronger relations:

$$T(\rho)(t) = T(\rho) + \max\{0, E(\rho) - t\}; \tag{7}$$

$$TW(\rho)(t) = TW(\rho) + \max\{0, t - L(\rho)\}. \tag{8}$$

Relation 8 is used to calculate effective TW violation due to later departing from the depot.

We introduce $R(\rho)$ as the largest release date of customers served in ρ. We define $R(v_i) = R_{v_i}$ and it holds

$$R(\rho_1 \oplus \rho_2) = \max\{R(\rho_1), R(\rho_2)\} \tag{9}$$

Thus, release dates, do not introduce any complexity into the moves evaluation scheme introduced in Section 4.2.

It remains to consider the multi-trip aspect. The idea is to consider each route σ_i as it would start at $\mathcal{T}^{\sigma_i} = 0$ and calculate variation in TW violations and time completion based on the effective starting time of the trip itself, i.e., considering $\mathcal{T}^{\sigma_i} = \max\left\{R(\sigma_i), \mathcal{T}^{\sigma_{i-1}} + \tau_{\sigma_{i-1}}\right\}$. Given a journey $\Sigma = (\sigma_1 \oplus \cdots \oplus \sigma_n)$, Equation 8 can be used to calculate $TW(\sigma_i)$ for all the trips in Σ, while $\mathcal{T}^{\sigma_{i+1}}$ can be calculated as

$$t = \begin{cases} E(\sigma_i) + T(\sigma_i) - TW(\sigma_i) & \text{if} \quad \mathcal{T}^{\sigma_i} \leq E(\sigma_i), \\ \mathcal{T}^{\sigma_i} + T(\sigma_i) - TW(\sigma_i) & \text{if} \quad E(\sigma_i) < \mathcal{T}^{\sigma_i} \leq L(\sigma_i), \\ L(\sigma_i) + T(\sigma_i) - TW(\sigma_i) & \text{if} \quad L(\sigma_i) < \mathcal{T}^{\sigma_i}. \end{cases} \tag{10}$$

$$\mathcal{T}^{\sigma_{i+1}} = \max\{t, R(\sigma_{i+1})\} \tag{11}$$

The effect of a move on trip $\sigma_i \in \Sigma$ can be propagated to each of the following routes by means of a variation in completion time and in TW penalization. Using Equations 7 and 8 this variation can be locally calculated in constant time. Complete calculation requires $O(\Sigma_{\max})$, where $\Sigma_{\max}$ indicates the maximum numbers of trips among all journeys.

4.4 Repair Procedure

Infeasible chromosomes are *repaired* applying LS with penalization factors multiplied by 10 depending on the nature of the infeasibility. If the resulting chromosome is still infeasible, λ and/or θ are multiplied again by 10 and LS is re-applied.

4.5 A split algorithm for the MTVRPTWR

4.5.1 Auxiliary graph construction

The splitting procedure *AdSplit*, is an adaptation of the procedure proposed by Prins in [11]. It works on an auxiliary graph $H = (V', A')$. V' contains $N + 1$ nodes indexed from 0 to N. Arc (i, j), $i < j$, represents a trip σ_{i+1}^{j} serving customers from Ψ_{i+1} to Ψ_j in the order they are in a given chromosome Ψ, i.e., $\sigma_{i+1}^{j} = (0, \Psi_{i+1}, \ldots, \Psi_j, 0)$. Since we are looking for feasible solutions, we could consider only feasible trips with the corresponding routing cost c_{ij} associated with arc (i, j), i.e.,

$$c_{ij} = T(\sigma_{i+1}^{j})$$

The best solution associated with chromosome Ψ would be represented by the path that goes from node zero to the node Ψ_N respecting time and capacity constraints.

In fact, we do not have the guarantee that such a feasible path (and then such a feasible solution) exists. We, then, allow infeasibility with respect to time and load constraints and we associate with each arc (i, j) the following cost

$$c_{ij} = T(\sigma_{i+1}^{j}) + \theta TW(\sigma_{i+1}^{j}) + \lambda \max\{0, D(\sigma_{i+1}^{j}) - Q\}. \tag{12}$$

It is noteworthy that the arc costs do not take into account the position of the trip in the journey, but it is the (penalized) cost of the trip as it would be performed as first (i.e., at $t = 0$). Therefore, the (contingent) TW violation due to later departure is not taken into account. As a consequence, considering the arcs that form the shortest path on H to construct the solution (as in the original procedure in Prins [11] for the VRP and in Cattaruzza et al. [4] for the MTVRP) could provide a solution with a cost much higher than the best one. We then propose a labelling procedure (explained in Section 4.5.2) that obtains a solution for the MTVRPTWR starting from the graph H constructed on Ψ.

4.5.2 Assignment of trips to vehicles

In the MTVRPTWR 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 routes leave the depot and this aspect cannot be considered by the constant costs associated with arcs on H (Section 4.5.1).

We propose the following labelling procedure that enumerates all the possible paths from node 0 to N in Ψ and construct the best solution associated with the label with lower cost on node N.

Starting from node 0, labels are progressively extended along the graph. Each label $\mathcal{L}$ has $M + 3$ fields: the first M fields store vehicle travel times in decreasing order, the $(M + 1)^{\text{th}}$ field memorizes the total load infeasibility, the $(M + 2)^{\text{th}}$ the predecessor node, and the last field keeps the cost of the partial solution evaluated using Equation 1 and equivalent to the cost $c(\mathcal{L})$ of label $\mathcal{L}$. When extending a label, M new labels are constructed, one for each possible allocation of the new trip to a vehicle.

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

To speed up the procedure, dominated labels, accordingly with the following *dominance rule*, are discarded: let $\mathcal{L}^1$ and $\mathcal{L}^2$ be two labels associated with the same node $i \in V'$. $\mathcal{L}^1$ *strongly* dominates $\mathcal{L}^2$ if and only if

$$c(\mathcal{L}^1) + \theta \sum_{j=1}^{M} \delta_j(\mathcal{L}^1, \mathcal{L}^2) \leq c(\mathcal{L}^2) \tag{13}$$

where $c(\mathcal{L})$ is the cost associated with label $\mathcal{L}$, $\delta_j(\mathcal{L}^1, \mathcal{L}^2) = \max\{0, T_j(\mathcal{L}^1) - T_j(\mathcal{L}^2)\}$ and $T_j(\mathcal{L})$ is the (partial) travel time of vehicle j associated with label $\mathcal{L}$. Roughly speaking, given two labels $\mathcal{L}^1$ and $\mathcal{L}^2$, extending $\mathcal{L}^1$ TW violation is penalized as much as possible while it is not extending $\mathcal{L}^2$ in the same way. If Inequality 13 holds, $\mathcal{L}^2$ cannot be extended in a better way than $\mathcal{L}^1$, then, $\mathcal{L}^2$ is eliminated.

Preliminary computational experiments (Section 5) show the time inefficiency of the procedure. For this reason a heuristic version of the dominance rule is proposed and a parameter $\gamma \geq 1$ is introduced. $\mathcal{L}^1$ *weakly* dominates $\mathcal{L}^2$ if and only if

$$c(\mathcal{L}^1) + \theta \sum_{j=1}^{M} \delta_j(\mathcal{L}^1, \mathcal{L}^2) \leq \gamma c(\mathcal{L}^2), \tag{14}$$

$$c(\mathcal{L}^1) \leq c(\mathcal{L}^2). \tag{15}$$

It is noteworthy that for $\gamma = 1$, the weak dominance rule is equivalent to the strong version. When $\gamma > 1$, Inequality 14 is easily satisfied and a larger number of labels can be eliminated. Condition 15 is added because when $\gamma > 1$ label $\mathcal{L}^2$ can be dominated by label $\mathcal{L}^1$ even if $c(\mathcal{L}^2) < c(\mathcal{L}^1)$. This cannot happen if $\gamma = 1$. 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 accordingly with the number of the label associated with each node. Precisely the following scheme is adopted:

$$\gamma = \begin{cases} \gamma + \frac{|\mathcal{L}_i|}{1000\mathcal{L}_{\text{threshold}}} & \text{if } |\mathcal{L}_i| > \mathcal{L}_{\text{threshold}} \\ \gamma - \frac{\mathcal{L}_{\text{threshold}}}{1000|\mathcal{L}_i|} & \text{if } |\mathcal{L}_i| < \mathcal{L}_{\text{threshold}} \end{cases} \tag{16}$$

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 Equation 16 is applied, γ is lower than 1, it is set back to 1. This is natural, since for $\gamma < 1$ dominated labels would be kept. The lower $\mathcal{L}_{\text{threshold}}$ the quicker is the procedure and the poorer is the quality of the solution obtained.

5 Discussion

In this section the computational results obtained are discussed. First of all we explain how we generated a set of 19 instances for the MTVRPTWR, since to the best of our knowledge there is not a benchmark of instances available in the literature for the problem. The value of the $\mathcal{L}_{\text{threshold}}$ is determined (Section 5.2). Finally results are presented (Sections 5.3- 5.4).

5.1 Instances generation

We define a generator of instances for the MTVRPTWR. The time horizon T_H is randomly drawn from $[500, 520]$ and is represented by a TW $[0, T_H]$ associated with the depot. The capacity Q of each vehicle is fixed to 100. The service time at the depot S_0 is 20, while each customer i requires a service time $S_i = 5$. Clients and the depot are located on a $[0, 50] \times [0, 50]$ squared Cartesian system portion. The depot is centrally located. For each customer, its abscissa and ordinate are randomly drawn in the interval $[0, 50]$. Given two customers i and j, the travel time T_{ij} is the rounded Euclidean distance between them. Customer's request is randomly drawn in the interval $[5, 15]$. The Clarke and Wright heuristic (Clarke and Wright [5]) is then applied to these data and a VRP solution is obtained. Trips σ obtained are concatenated as long as the journey respects depot's TW. Corresponding $\mathcal{T}^\sigma$ are calculated. If a trip cannot be assigned to any vehicle, a new journey is initialized with it. M is fixed to the number of vehicles which are needed to assign each trip to a journey. For each customer i, two dates are calculated: the starting of the service $\underline{s}_i$ time when the vehicle leaves the depot at $t = 0$ and $\overline{s}_i$ when the vehicle leaves the depot as late as possible respecting T_H. Then s_i is calculated as the average of $\underline{s}_i$ and $\overline{s}_i$. L_i is then calculated as follows:

- If $s_i < 240$
  - $L_i = 240$ with a probability of 0.7;
  - $L_i = 360$ with a probability of 0.2;
  - $L_i = 480$ with a probability of 0.1.
- If $240 \leq s_i < 360$
  - $L_i = 360$ with a probability of 0.6;
  - $L_i = 480$ with a probability of 0.4.
- Otherwise $L_i = 480$.

The release dates R_i are determined as follows:

- $R_i = 0$ with a probability of 0.5.
- Otherwise R_i is randomly drawn from $[0, R_\sigma]$, where $R_\sigma = \min_{j \in \sigma}\{\mathcal{T}^\sigma - s_j, L_j - s_j\}$ and $i \in \sigma$.

Finally $E_i = R_i + S_0 + T_{0i}$. A set of 19 instances is then created with 20 to 200 customers and 1 to 7 vehicles.

Strong		Weak							
		$\mathcal{L}_{\text{threshold}} = 20$		$\mathcal{L}_{\text{threshold}} = 10$		$\mathcal{L}_{\text{threshold}} = 5$		$\mathcal{L}_{\text{threshold}} = 1$	
Cost	Time	Cost	Time	Cost	Time	Cost	Time	Cost	Time
1138.94	193.6	0.13%	-99.74%	0.19%	-99.87%	0.21%	-99.94%	7.67%	-99.99%
			0.51		0.23		0.09		0.01

Table 1: Comparison of dominance rules: average results obtained by *AdSplit* procedure on 60 chromosome for an instance with $M = 6$ and $N = 150$ (times in seconds)

M	N	Best TS cost	Average TS % gap	HGA % gap	Best known cost
1	20	203	0.0	0.0	203
2	30	309	0.0	0.0	309
2	40	347	0.0	0.0	347
2	50	384	2.9	0.0	384
3	60	491	1.6	0.0	491
3	70	469	1.9	0.0	469
3	80	571	1.9	-0.4	569
4	90	570	5.8	0.9	570
4	100	642	4.5	-0.2	641
4	110	786	2.3	-1.9	771
5	120	755	5.2	-0.7	750
5	130	827	3.1	-1.9	811
5	140	952	4.0	-4.5	909
6	150	971	7.5	-0.9	962
6	160	1028	5.1	-1.9	1008
6	170	1033	3.1	2.0	1033
7	180	1197	5.1	-7.5	1107
7	190	1208	7.4	-3.6	1164
7	200	1245	6.7	-2.6	1213

Table 2: Results on the instances for the MTVRPTWR

5.2 Determination of $\mathcal{L}_{\text{threshold}}$

To determine the value of $\mathcal{L}_{\text{threshold}}$, 60 chromosomes for an instance with $N = 150$ and $M = 6$ are randomly generated. Then, they are evaluated by the *AdSplit* procedure with the use of the strong dominance rule and the weak dominance rule with different values of $\mathcal{L}_{\text{threshold}}$. Results are reported in Table 1. It can be noticed that using the strong dominance rule more than 3 minutes are averagely needed to split a chromosome. This justifies the introduction of the weak dominance rule. Still from Table 1 it can be noticed that decreasing $\mathcal{L}_{\text{threshold}}$, the procedure is quicker but the quality of the obtained solutions is slightly reduced. It has been decided to set $\mathcal{L}_{\text{threshold}} = 5$.

5.3 Comparison with a classical Tabu Search algorithm

The procedure is run once over all instances for 5 minutes. Results are compared with those obtained by a Tabu Search (TS) algorithm that we developed to obtain reference values. The TS makes use of the classical neighborhoods insert and swap. Seven structures are considered combining them differently. The TS is run 6 times on each instance with each one of the different structures for a total of 42 times (each stopped after 5 minutes of computation). Results are reported in Table 2.

The first two columns report the number of vehicles and customers on each instance. Column *Best TS* reports the cost of the best solution obtained by the TS procedure over the 42 runs, while column *Average TS* reports the gap of the average cost of solution obtained over the 42 runs with respect to the

Instance	opt	HGA	CPU Time
C201_2_25	380.8	387.7	300.0
C202_2_25	368.6	368.6	2.1
C203_2_25	361.7	361.7	0.3
C204_2_25	358.8	358.8	0.0
C205_2_25	377.2	377.2	0.0
C206_2_25	367.2	367.2	0.2
C207_2_25	359.1	359.1	0.3
C208_2_25	360.9	360.9	15.5
R201_2_25	554.6	554.6	0.1
R202_2_25	485.0	485.0	1.2
R203_2_25	444.2	444.2	0.3
R204_2_25	407.5	407.5	8.0
R205_2_25	448.4	448.4	0.0
R206_2_25	413.9	413.9	0.5
R207_2_25	400.1	400.1	0.3
R208_2_25	394.3	394.3	2.7
R209_2_25	418.3	418.3	0.1
R210_2_25	448.3	448.3	117.4
R211_2_25	400.1	400.1	0.2
RC201_2_25	660.0	660.0	27.8
RC202_2_25	596.8	596.8	1.2
RC203_2_25	530.1	530.1	0.0
RC205_2_25	605.3	605.3	0.9
RC206_2_25	575.1	575.1	27.1
RC207_2_25	528.2	528.2	0.1
R201_4_50	909.8	922.7	300.0
R202_4_50	816.0	816.0	69.6
R205_4_50	807.3	816.2	300.0
RC201_4_50	1096.6	1097.6	300.0
RC202_4_50	1001.6	1038.6	300.0

Table 3: Comparison with results of Hernandez et al. [9]

value in *Best TS*. Column *HGA* reports the cost of the solution obtained by the HGA procedure run just once. Column Best known reports the cost of the best known solution. HGA is better than TS on 11 instances and provides equivalent solutions on 6 instances.

5.4 Comparison with Hernandez et al. [9]

Hernandez et al. [9] propose an exact method for the MTVRPTW and are able to solve 30 out of 54 instances they construct from Solomon's instances for the VRPTW. We run our algorithm on those instances ($R_i = 0$ for all i). Results are reported in Table 3.

The first column reports the instance name ON_M_N, where ON is Solomon's instance original name. The second column reports the optimal value, the third the cost of the solution found by HGA, while the last column reports the CPU time (in seconds). The procedure is stopped when the optimal solution is found or after 5 minutes. HGA finds the optimal value 25 out of 30 times missing it just once for instances with $M = 2$ and $N = 25$. The average gap from the optimal value is 0.4%.

6 Conclusion

In this paper, we introduced the Multi Trip Vehicle Routing Problem With Time Windows and Release Dates. This problem is particularly interesting in the City Logistics context. We proposed a hybrid GA based on a new split procedure to face it. A set of instances has been introduced. Comparison with optimal solution of Hernandez et al. [9] proves the efficiency of our algorithm.

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] D. Cattaruzza, N. Absi, D. Feillet, and T. Vidal. A memetic algorithm for the multi trip vehicle routing problem. Technical Report 2012, Ecole des Mines de Saint-Etienne, 2012.

[5] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[6] M. Drexl. Synchronization in vehicle routing – a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46:297–316, 2012.

[7] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44(3):155–170, 2004.

[8] 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. Technical report, laboratoire d'Informatique de Robotique et de Microlectronique de Montpellier (LIRMM), 2011.

[9] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud. An exact algorithm to solve the multi-trip vehicle routing problem with time windows. *Elsevier*, 00(0):1–28, 2012.

[10] 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.

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

[12] 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.

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