Adaptive large neighborhood search for multi-trip vehicle routing with time windows

Véronique François[1], Yasemin Arda[1], and Yves Crama[1]

[1]HEC Liège - Management School of the University of Liège

Abstract

We consider a multi-trip vehicle routing problem with time windows (MTVRPTW), in which each vehicle can perform several trips during its working shift. This problem is especially relevant in the context of city logistics. Heuristic solution methods for multi-trip vehicle routing problems often separate routing and assignment phases in order to create trips and then assign them to the available vehicles. We show that this approach is outperformed by an integrated solution method in the presence of time windows. We use an automatic configuration tool to obtain efficient and contextualized implementations of our solution methods. We provide suitable instances for the MTVRPTW as well as an instance generator. Also, we discuss the relevance of two objective functions: the total duration and the total travel time. When minimizing the travel time, large increases of waiting time are incurred, which is not realistic in practice.

1 Introduction

In a multi-trip vehicle routing problem (MTVRP), each vehicle performs a set of trips, whose total duration, corresponding to the vehicle travel time, is limited by the length of the planning horizon. In a multi-trip vehicle routing problem with time windows (MTVRPTW), customers must be served within a specified time window and the duration of a set of trips assigned to a vehicle is usually bounded by the length of the depot's time window which corresponds to the planning horizon. These two problems are common in city distribution systems (Cattaruzza et al., 2017). In urban areas, small vehicles are encouraged for environmental reasons and, in most cases, required due to accessibility restrictions, limiting the loading capacity of the vehicles. Moreover, travel times between delivery points are short compared to the planning horizon, allowing vehicles to return to the depot several times for reloading. In this context, multiple trips arise naturally. They are also common when products are ready for delivery at different

points in time to be distributed on the same day. This is for example the case in nuclear medicine delivery (Lee et al., 2014) and home chemotherapy systems (Kergosien et al., 2017), since products are highly perishable and have to be delivered as early as possible after their production.

In this article, we study an MTVRPTW with a given fleet size and driver shifts shorter than the planning horizon. This operational problem is common in many distribution systems with a one-day planning period. Indeed, to increase the satisfaction of customers, or to avoid traffic jams, some deliveries have to be performed in the early morning, and some others in the late evening, implying that each driver can only cover a portion of the planning period. The driver shifts we consider have a limited duration and are not fixed in time, i.e., the start time of each shift is a decision variable.

Additionally, we minimize the total working duration, which includes the travel time, the waiting time as well as loading and service times, instead of the most employed objective function, i.e., the total travel time. Indeed, waiting times may induce direct or indirect costs linked to specialized equipment. In the particular case of refrigerated urban distribution, the negative impact on the environment of logistics activities increases because of the refrigeration equipment. The usage cost of refrigerated vehicles and their environmental impact are dependent on the duration of the equipment use and not only on the travel time. Moreover, waiting times may have a significant impact on the transportation costs, especially because of the importance of driver wages, which are dependent on national regulations, while being always significant. According to Belgian and French national transporter associations, when a carrier transports goods with small trucks as the ones used in city logistics, the driver wages represent more or less 40 percent of the total transportation costs, which also include costs related to routing, maintenance, transport administration, etc. (Union Professionnelle du Transport et de la Logistique, 2012; Comité National Routier, 2017). Depending on the context, time spent waiting might be used to perform other tasks, implying an opportunity cost for the company. Home chemotherapy with treatments administered by hospital nurses is an example implying qualified workers whose time is very valuable. Time windows imposed by the schedule of active patients or by the medical requirements may cause waiting time for nurses, preventing them from performing additional tasks at the hospital after their home care journey. Even in industrial contexts, when the fleet size is determined at a tactical level, the fleet may be oversized for particular operational problems encountered on specific days. Drivers may then use their extra time for logistics operations inside the depot instead of waiting on the road, depending on their contract. Also, the repartition of the customer time windows along the day may necessitate more drivers during short intervals. For all the mentioned reasons, we consider the minimization of the total working duration as a very relevant objective, especially given that in many cases large amounts of waiting time may be saved at the cost of a comparatively small travel time increase, as shown later in this work.

1.1 Problem Description

In the MTVRPTW considered in this work, a fleet of m identical capacitated vehicles based at a unique depot serves a set of geographically scattered customers. The problem may be defined on a complete graph $G(V, E)$, where the vertex set $V = \{0, 1, ..., n\}$ represents the depot (0) and the customers. Each edge (i, j) has a weight t_{ij} equal to the travel time between vertices i and j. The demand d_i of each customer $i \in V \setminus \{0\}$ must be satisfied, while its service has a duration s_i and begins compulsorily within a given period of time, i.e., a hard time window $[a_i, b_i]$. If the vehicle arrives earlier than the beginning of the time window, it has to wait. Arriving after the end of the time window is prohibited.

A trip is defined by a sequence of customer visits that starts and ends at the depot. The subset of customers served in a trip r is denoted by V_r. At the beginning of trip r, a loading time $l_r = \gamma \sum_{i \in V_r} s_i$ is incurred where $\gamma \geq 0$ is a constant, called loading factor. Each vehicle k is allowed to perform a tour $\mathcal{T}_k$ consisting of a sequence of one or more trips that do not overlap in time. Each vehicle must start and end its tour within the planning horizon defined by $[a_0, b_0]$. Moreover, the duration of a tour cannot exceed an upper bound D^{max} that represents the maximum duration of a driver shift.

A solution is defined by a set of tours that ensures that each customer is visited exactly once during its time window, and that the vehicle capacities are respected for each trip. As further explained in Section 2.2, the exact schedule associated to a solution is not explicitly defined. For each tour $\mathcal{T}_k$, there exists a start time interval $[E_{\mathcal{T}_k}, L_{\mathcal{T}_k}]$ at the depot that guarantees a minimum tour duration, while ensuring the feasibility of the tour schedule if any such schedule exists. An explicit tour schedule may be computed by choosing any start time within this interval.

The objective of the MTVRPTW is to find a set of tours that minimizes the total duration $\sum_{k=1,...,m} D_k$ where D_k is the duration of tour $\mathcal{T}_k$. The total duration includes service times, loading times, travel times, and waiting times. The first two terms are constant for a given instance and do not influence the value of the objective function, but they impact the satisfaction of time-related constraints. The last two terms need to be minimized and usually tend to compete with each other.

1.2 Literature Review

The MTVRP has been introduced by Fleischmann (1990) and the related literature has been slowly growing ever since. Most references that address the original MTVRP, where each vehicle can perform a set of trips in a limited period of time, are using VRP heuristics to create trips in combination with bin packing techniques to assign trips to vehicles (e.g. Fleischmann, 1990; Taillard et al., 1996; Olivera and Viera, 2007; Cattaruzza et al., 2014). François et al. (2016) studied how a heuristic method decomposing the routing and the packing aspects of an MTVRP competes with an integrated method using multi-trip local search operators. The integrated approach was promising but slightly outperformed by the routing-packing decomposition. However, as mentioned in Cattaruzza et al. (2016b) and François et al.

(2016), when studying MTVRP variants including time windows, heterogeneous fleets, and other side constraints, it becomes more difficult for a routing-packing method to find a suitable assignment of a given set of trips to the available vehicles.

As shown in the survey of Cattaruzza et al. (2016b), different MTVRPTW variants have been treated with both exact and heuristic methods. In the following, we concentrate on references sharing similar characteristics with our problem.

The MTVRPTW variant that we consider has two additional features: service-dependent loading times and a limited tour duration. While service-dependent loading times are commonly encountered in the MTVRPTW context, we are only aware of two references that consider a limited tour duration shorter than the planning horizon, namely Battarra et al. (2009) and Despaux and Basterrech (2016). This constraint should not be mistaken for the limited trip duration introduced in Azi et al. (2014), and considered in several references (e.g., Azi et al., 2014; Hernandez et al., 2014; Wang et al., 2014), which is specifically relevant when delivering highly perishable goods. Indeed, limiting the trip duration favors the creation of multiple trips by forcing the vehicle to return periodically to the depot. However, in an MTVRPTW without limited trip duration as the one we consider, the multiple trips arise naturally because of the characteristics of the underlying instances in terms of travel times and quantities to be delivered.

Battarra et al. (2009) consider an MTVRPTW with multiple commodities and a limited tour duration, called *spread time*. The authors seek to minimize the fleet size and then the total travel distance. They develop a two-phase heuristic method: a VRPTW heuristic creates trips that are later combined into tours by a greedy procedure. The method iterates through these two phases and penalty mechanisms embedded in the VRPTW heuristic discourage the creation of trip sets that are likely to require a high number of vehicles. Despaux and Basterrech (2016) study an MTVRPTW with heterogeneous fleet and a limited tour duration where a combination of travel costs and fixed vehicle costs have to be minimized. A simulated annealing metaheuristic integrating several neighborhood structures is used to solve the problem. Most moves modify the assignment of customers to trips. One of the neighborhoods modifies the assignment of trips to vehicles.

Cattaruzza et al. (2016a) consider an MTVRPTW with release dates where each order associated to a given customer arrives at the depot at a precise time within the planning period, implying a customer-dependent release date on the deliveries. A hybrid genetic algorithm is developed where each chromosome is a sequence of customers. The split procedure of (Prins, 2004) is modified to create a multi-trip solution from each chromosome and local search procedures improve the created solutions. To the best of our knowledge, the only reference that addresses the MTVRPTW exactly without imposing a limit on the trip duration is the article of Hernandez et al. (2016). The authors compare two set covering formulations for solving the MTVRPTW by means of branch-and-price algorithms. The variables are tours in the

first formulation, and they are trips in the second. Promising results are obtained on instances with 25 customers and the authors notice that the branch-and-price algorithm based on the second formulation is globally the best choice to achieve feasibility on the test instances.

The objective function commonly studied in the context of MTVRPTW is the total travel distance or travel time, which most authors consider to be equivalent. Profits are introduced when it is not mandatory to visit all the customers (e.g., Azi et al., 2014; Wang et al., 2014). In this work however, we minimize the total duration. As noted in Savelsbergh (1992), this is a much more realistic objective function in the presence of time windows. Indeed, waiting times can have a significant impact on the duration. In Savelsbergh (1992), concatenation equations are developed to merge feasible paths efficiently in order to evaluate the duration change induced by edge exchange moves. These concatenation equations have been broadly used in the context of VRPTW. Recently, Vidal et al. (2015) proposed new concatenation techniques that allow evaluating efficiently the concatenation of infeasible paths where time window violations are penalized using the concept of *time wrap* developed by Nagata et al. (2010).

1.3 Contribution

As mentioned in Section 1.2, when using routing-packing decomposition approaches, the presence of time windows complicates the assignment of a given set of trips to available vehicles, since generated trips usually have time overlaps. Moreover, a feasible trip may become infeasible if shifted in time for assignment purposes. Even when a feasible assignment is found, it may not be the one that would minimize the total duration since the duration of a feasible trip depends on its start time and may thus change if assigned differently. The main objective of this article is to compare the efficiency of an integrated approach with the one of a routing-packing decomposition approach, in the presence of time windows. To obtain a meaningful comparison, we develop two heuristic methods based on the same framework, i.e., the adaptive large neighborhood search (ALNS) of Ropke and Pisinger (2006a) and Pisinger and Ropke (2007). The first one, named ALNSM, iteratively modifies vehicle tours using multi-trip operators. The second one, named ALNSP, iteratively modifies trips and then assigns them to the available vehicles. In François et al. (2016), both strategies have been compared on the classical MTVRP, without time windows. The integrated approach proves interesting but does not outperform the routing-packing decomposition approach for this less constrained problem. In this work, we extend both methods to treat the MTVRPTW while allowing the evaluation of solutions that are not feasible with respect to time-related constraints. For that purpose, we embed the recent concatenation techniques of Vidal et al. (2015) to efficiently evaluate ALNS moves. We show how to adapt these concatenation techniques to take into account service dependent loading times in a multi-trip context. We also use these concatenation operations within the assignment phase of the ALNSP, which significantly differs from the non-constrained case presented in François et al. (2016).

Through extensive numerical analyses, we assess the efficiency of our algorithms by comparing their results with those of existing methods. A significant contribution of this work is to provide evidence that the multi-trip operators are very efficient in the presence of time windows. Hence, taking advantage of the complete solution structure, instead of decomposing it, is shown to be a very effective strategy. This contribution is not only important in the context of time windows. Indeed, when constraints bind customers to a limited set of vehicles, e.g., when certain drivers are preferred by customers or when fleet accessibility restrictions apply, routing-packing decomposition approaches become challenging to implement since trips have to be created without restraining their assignment options. On the contrary, an integrated solution method, where moves are applied directly on a set of multi-trips, is much more intuitive and avoids the drawbacks of the routing-packing decomposition strategy.

Additionally, we provide new instances for the MTVRPTW, whose characteristics naturally favor the occurrence of multiple trips. Indeed, the instances considered by previous researchers tend to produce very few (either one or two) trips per vehicle, and hence do not display the full complexity of multi-trip situations encountered in practice. The instances that we create give rise, on average, to more trips than these earlier benchmarks and hence, can be viewed as more realistic. The instance generator is made available to the community.

We also analyze the relevance of two different objective functions, namely: the total duration and the total travel time. This results in an important conclusion stating that accepting a small travel time increase may result in huge savings in terms of waiting time, and suggests that the minimization of the total duration should be further studied in the presence of time windows. A consequence of considering the minimization of the total duration as an objective function is that the vehicle start times are decision variables. In this work in particular, vehicle start times are considered as decision variables even when the objective is to minimize the total travel time. Indeed, departing at the beginning of the planning horizon may prevent the satisfaction of the maximum allowed duration constraints.

In a classical ALNS algorithm, many numerical parameters are needed. Besides, the flexibility of our methods in terms of algorithmic design is augmented by means of binary parameters. The parameter values of both methods are determined using an automatic configuration tool, `irace` (López-Ibáñez et al., 2016). According to the configurations provided by `irace`, the parameters take different values depending on the method characteristics and on the objective function to be minimized. This shows the importance of using an automatic configuration method to obtain a contextual implementation.

Throughout this paper, a particular attention is drawn to methodological contributions. Indeed, we challenge the relevance of a commonly used objective function, we question the characteristics of former training and testing instances, and we carefully configure our solution methods by means of a dedicated tool. We believe that these methodological aspects are at least as important as the numerical results.

2 Solution Methods

The ALNSM and ALNSP solution methods are detailed in Sections 2.3 and 2.4 respectively. Both methods are based on the ALNS framework that relies on a set of removal heuristics and a set of insertion heuristics, which iteratively destroy and repair solutions. Removal (resp. insertion) heuristics differ in the criterion used to select the customers to be removed (resp. to be inserted). An adaptive weight is assigned to each heuristic and influences its selection probability at each iteration. Our set of heuristics is described in Section 2.5. When working on a set of trips, i.e., on a partial VRPTW solution, customers are removed and reinserted between any two vertices of any trip. When considering a set of tours, i.e., a partial MTVRPTW solution, removing and inserting copies of the depot is allowed in order to facilitate changes in the tour structures as explained in Section 2.1.

Both algorithms depend on several numerical and boolean parameters. The configuration phase, in which parameter values are determined, is further described in Section 3.1.

2.1 Specific Local Search Operators

Multi-trip versions of VRP operators proposed in François et al. (2016) are designed to modify vehicle tours without decomposing them into their constituting trips. That is, each vehicle tour is described as a unique sequence of vertices that starts at an origin depot, continues with the visited customers and internal depots where reloading operations are performed, and ends at a destination depot. These specific operators, used in the ALNSM, are briefly sketched here. The details of the exact use of the removal and insertion operators is provided in Section 2.5.

Removal and Insertion Operators. When a customer is inserted into a partial MTVRPTW solution, four insertion schemes are considered:

1. Insert the customer.
2. Insert an internal depot and then the customer.
3. Insert the customer and then an internal depot.
4. Insert a new trip between two consecutive customers; this new trip visits only the customer to insert.

Removal operators remove one customer at a time. If there are two consecutive depots in a tour after removing a customer, one of these depots is also removed. An operator allows merging two consecutive trips of a tour by removing the depot between them. Mergers are recursively performed if needed.

Improvement Operators. We consider two types of improvement moves, i.e., the exchange or relocation of customer sequences. These operators are used on a multi-trip representation of the tour of each vehicle, as explained above, treating customers and internal depots in the same way.

2.2 Search Space

In order to allow visiting infeasible solutions, both our ALNS implementations work on a relaxed version of the MTVRPTW, called the r-MTVRPTW, in which the time-related constraints, i.e., the maximum tour duration and the time windows, are relaxed. In the following, we denote an MTVRPTW solution by $\mathcal{S}$, and a r-MTVRPTW solution by $\hat{\mathcal{S}}$. As it is the case for MTVRPTW solutions, the schedule of r-MTVRPTW solutions is not explicitly defined. It is not needed to evaluate time window and tour duration violations.

In order to evaluate the cost of violating time windows, we adopt the time window violation scheme of Nagata et al. (2010). If a vehicle arrives late at a customer or at the depot, we pretend that it is allowed to travel back in time to arrive at the customer or at the depot just at the end of the corresponding time window. That is, the service start time of a vehicle at a customer i fictively never exceeds b_i. The time wrap usage (Nagata et al., 2010) denotes the amount of time units traveled back in time. The total amount of time wrap necessary to serve all the customers on time in a given r-MTVRPTW solution $\hat{\mathcal{S}}$ is recorded as $TW(\hat{\mathcal{S}})$.

If at least one vehicle of a r-MTVRPTW solution $\hat{\mathcal{S}}$ travels for a duration that exceeds D^{max}, then $\hat{\mathcal{S}}$ contains overtime. The overtime of vehicle k, denoted by OT_k, is defined as $OT_k = \max\{0, D_k - D^{max}\}$. The total overtime of $\hat{\mathcal{S}}$ is defined as $OT(\hat{\mathcal{S}}) = \sum_{k=1,\dots,m} OT_k$.

When guiding the search, the cost of a candidate solution $\hat{\mathcal{S}}$ is defined as $C(\hat{\mathcal{S}}) = \sum_k D_k + \alpha(TW(\hat{\mathcal{S}}) + OT(\hat{\mathcal{S}}))$, where $\alpha \in [\alpha_{min}, \alpha_{max}]$ is an adaptive parameter. The value of α is initialized to α_{min} and a parameter $\mu \geq 1$ controls its variation as described in Olivera and Viera (2007). Whenever an accepted r-MTVRPTW solution contains overtime or time wrap, the value of α is set to $\min\{\alpha\mu, \alpha_{max}\}$ to focus on reducing infeasibility. Else, the value of α is set to $\max\{\alpha/\mu, \alpha_{min}\}$ to encourage visiting infeasible solutions. After ξ iterations, the value of α is reset to α_{min} to prevent it from remaining stuck at its maximum value α_{max}. The values of α_{min}, α_{max}, μ, and ξ are determined during the configuration phase.

For the purpose of detecting improvements or new best solutions, the cost measure of a r-MTVRPTW solution becomes $\bar{C}(\hat{\mathcal{S}}) = \sum_k D_k + M(TW(\hat{\mathcal{S}}) + OT(\hat{\mathcal{S}}))$, where M is a very large number. This implies that, when performing a comparison between two r-MTVRPTW solutions, their total durations are only taken into account if they are both feasible MTVRPTW solutions or if they have the same total amount of time-related infeasibilities.

Vidal et al. (2015) propose concatenation equations to evaluate vertex-based or edge-based moves in $\mathcal{O}(1)$, even though infeasible solutions are visited during the course of the algorithm. Each move applied to a solution is described in terms of path concatenations, where each path is a sequence of vertices. Four measures describe time-related aspects of any path τ: the minimum duration $D(\tau)$ and time wrap usage $TW(\tau)$ necessary to serve all the vertices of τ, as well as the earliest and latest start times at the first

vertex, $\tau(1)$, of the path, $E(\tau)$ and $L(\tau)$. The start time interval $[E(\tau), L(\tau)]$ is the one that minimizes the duration and the amount of time wrap of the considered path.

In this work, contrary to the case described in Vidal et al. (2015), the values associated with a given path depend on the considered concatenation operation. This is due to the service-dependent loading times that impact depots. For a path τ containing a single customer i, the four measures $D(\tau)$, $TW(\tau)$, $E(\tau)$, and $L(\tau)$ are respectively equal to s_i, 0, a_i, and b_i. These values are only dependent on the characteristics of the customer itself. However, for a path containing only a depot, the same four measures are respectively l_r, 0, a_0, and b_0, where l_r is the loading time of the trip r following the depot in the considered concatenation operation. To better understand the impact of this dependency, an example follows the concatenation equations below.

Let τ_1 and τ_2 be two paths, the equations of Vidal et al. (2015) reported below allow computing the four measures for $\tau_1 \oplus \tau_2$, i.e., the concatenation of τ_1 and τ_2.

$$\Delta = D(\tau_1) - TW(\tau_1) + t_{\tau_1(|\tau_1|),\tau_2(1)} \tag{1}$$

$$\Delta WT = \max\{E(\tau_2) - \Delta - L(\tau_1), 0\} \tag{2}$$

$$\Delta TW = \max\{E(\tau_1) + \Delta - L(\tau_2), 0\} \tag{3}$$

$$D(\tau_1 \oplus \tau_2) = D(\tau_1) + D(\tau_2) + t_{\tau_1(|\tau_1|),\tau_2(1)} + \Delta WT \tag{4}$$

$$E(\tau_1 \oplus \tau_2) = \max\{E(\tau_2) - \Delta, E(\tau_1)\} - \Delta WT \tag{5}$$

$$L(\tau_1 \oplus \tau_2) = \min\{L(\tau_2) - \Delta, L(\tau_1)\} + \Delta TW \tag{6}$$

$$TW(\tau_1 \oplus \tau_2) = TW(\tau_1) + TW(\tau_2) + \Delta TW \tag{7}$$

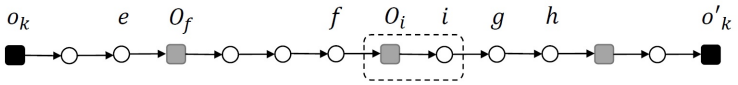
In the following, we detail how to maintain the values needed to evaluate moves in $\mathcal{O}(1)$ when using multi-trip operators in the context of service-dependent loading times. Let o_k be the origin depot of tour $\mathcal{T}_k$ and o'_k be its destination depot. For each vertex i in $\mathcal{T}_k$, let O_i and O'_i denote respectively the origin and the destination depot of the trip containing i. Note that o_k is equivalent to O_i if vertex i belongs to the first trip of $\mathcal{T}_k$. Similarly, o'_k is equivalent to O'_i if i belongs to the last trip of $\mathcal{T}_k$. When updating $\mathcal{T}_k$, for each i in $\mathcal{T}_k$, we maintain the four measures D, TW, E and L for four different paths:

- $\tau_{[o_k,i]}$, from the origin depot of $\mathcal{T}_k$ included to vertex i included,
- $\tau_{[i,o'_k]}$, from vertex i included to the destination depot of $\mathcal{T}_k$ included,
- $\tau_{(O_i,i]}$, from the depot preceding vertex i excluded to vertex i included,
- $\tau_{[i,O'_i)}$, from vertex i included to the depot following vertex i excluded.

Moreover, the cumulated service time $\bar{s}_i$ necessary to serve the customers up to vertex i on path $\tau_{(O_i,i]}$ is also recorded. The first three path types are used to evaluate insertion and removal moves. The fourth one

is needed for merge moves. Note that only the first two types of paths are maintained for internal depots. When inserting or removing a customer, the loading time at the preceding depot changes, implying that the path which precedes the inserted or removed customer must be decomposed as represented by the following figure.

Figure 1: Insertion move (scheme *2*) in a tour $\mathcal{T}_k$



In Figure 1, a move to be evaluated on a tour $\mathcal{T}_k$ is represented. The circles stand for the customers and the rounded squares are the depots. Internal depots are in light gray, while the origin and destination depots are in black. The insertion of customer i between customers f and g is evaluated with insertion scheme 2: an internal depot and customer i are inserted between customers f and g. The elements to insert are represented inside the dotted rectangle. After the considered insertion, the loading time at depot O_f changes. Thus, the time-related measures associated with path $\tau_{[o_k,f]}$ are not valid anymore. Consequently, the new values of $E(\mathcal{T}_k)$, $L(\mathcal{T}_k)$, $D(\mathcal{T}_k)$, and $TW(\mathcal{T}_k)$ must be calculated by concatenating six paths: $\mathcal{T}_k \leftarrow \tau_{[o_k,e]} \oplus O_f \oplus \tau_{(O_f,f]} \oplus O_i \oplus i \oplus \tau_{[g,o'_k]}$. Time-related measures for paths $\tau_{[o_k,e]}$, $\tau_{(O_f,f]}$ and $\tau_{[g,o'_k]}$ are known since they were maintained during former solution updates. The three other paths contain one single vertex and their time-related measures are set accordingly. Note however that $D(O_f)$ and $D(O_i)$ are not equal to 0 because the loading time at the depots depends on the customers to be served on the associated trip. Maintaining the cumulative service times up to date allows computing $D(O_f) = \gamma \bar{s}_f$ and $D(O_i) = \gamma(\bar{s}_h - \bar{s}_f + s_i)$ in $\mathcal{O}(1)$.

2.3 Adaptive Large Neighborhood Search with Multi-Trip Operators (ALNSM)

In this section, we describe the ingredients of the ALNSM algorithm, which explores the search space by means of multi-trip operators that destroy and recreate r-MTVRPTW solutions.

Internal depots may be removed if empty trips are created through removal heuristics. A merge operator is included by default within all removal heuristics: two consecutive trips are merged, i.e., the internal depot between them is removed, if their merger is feasible with respect to capacity constraints. A version without this merge operator is also implemented for each heuristic and a boolean parameter determines if these modified removal heuristics are included in the final algorithm design or not.

When repairing a solution with insertion heuristics, the four multi-trip insertion schemes of Section 2.1 are used. Since we do not allow the violation of capacity constraints, we first evaluate the insertion of a customer with scheme 1, which is always the least costly. If it is infeasible with respect to capacity

constraints, we evaluate both schemes 2 and 3 to determine the least costly one. Scheme 4 is evaluated only if all other schemes are infeasible with respect to capacity constraints since it is always the costliest option.

Initial Solution. An initial r-MTVRPTW solution is built by iteratively adding an unrouted customer at its best possible position using one of the four multi-trip insertion schemes. At each iteration of this greedy procedure, the customer to be added and the insertion scheme are chosen so as to minimize the cost increment (with $\alpha = \alpha_{min}$) of the partial solution while preserving feasibility with respect to capacity constraints.

Destroy-and-Repair. At each iteration of the ALNSM, the incumbent solution $\hat{\mathcal{S}}$ is destroyed using a removal heuristic randomly selected from a set H_{rem}, and then repaired with an insertion heuristic randomly selected from a set H_{ins}. A roulette wheel mechanism, derived from the one proposed by Ropke and Pisinger (2006a), aims at periodically adapting the selection probability of each heuristic. The heuristics that led to improved r-MTVRPTW solutions during previous iterations, or that favored diversification by producing solutions previously unvisited are rewarded, and their selection probability increases consequently. Reversely, unpromising heuristics are assigned lower selection probabilities. The implementation details of the roulette wheel mechanism are described in François et al. (2016) and summarized below.

Each removal and insertion heuristic is provided with a weight in $]0, 1[$, i.e., a selection probability. At the beginning of an algorithm, all the weights are the same. Each time choosing a given heuristic leads to an improved solution, a new best solution, or a previously unvisited solution, a reward is granted to this heuristic. After a certain number of iterations, called time segment, the heuristic weights are adapted to take into account their respective rewards, and a new time segment begins with modified heuristic weights. Four numerical parameters control the roulette wheel : σ_1 and σ_2 influence the rewards, ρ determines the persistence of information between time segments, and Θ is the length of time segments.

The number of customers to be removed and reinserted at a given iteration is controlled by an adaptive parameter q managed as follows. In the first iteration, and each time a solution is accepted, q is set to 1. If the candidate solution is rejected, q is set to $q + 1$. If several consecutive candidate solutions have been rejected and $q = q_{max}$, upon an additional rejection, q is set to q_{low}. The values of q_{max} and q_{low} are determined as functions of the instance size during the configuration phase.

Solution Acceptance. A simulated annealing framework is used to decide whether to accept a candidate solution $\hat{\mathcal{S}}'$ given the incumbent solution $\hat{\mathcal{S}}$. If $C(\hat{\mathcal{S}}') < C(\hat{\mathcal{S}})$, then $\hat{\mathcal{S}}'$ is accepted as the new incumbent. Else, $\Pr(\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}}') = \exp[-(C(\hat{\mathcal{S}}') - C(\hat{\mathcal{S}}))/\theta C(\hat{\mathcal{S}})]$, where θ is the temperature. At each iteration, θ is set to $\max\{\eta\theta, \theta_{min}\}$, where $\eta \in [0, 1]$ is the cooling factor. The minimum temperature is

defined as $\theta_{min} = \kappa\theta_0$ with $\kappa \in [0, 1]$.

Note that for the purpose of updating the roulette wheel parameters, improvements and new best solutions are detected using the modified cost measure $\bar{C}$ as explained in Section 2.2. Indeed, the cost measure $\hat{C}$ used to accept or reject new solutions does not depend only on the solution characteristics but also on the value of the adaptive parameter α during a given iteration. On the contrary, $\bar{C}$ allows a comparison that stays consistent throughout the course of the algorithm.

Post-Optimization. Once the maximum run time is reached, a variable neighborhood descent (VND, see Hansen and Mladenović (2001)) tentatively improves $\hat{\mathcal{S}}_{best}$. The maximum run time depends on the instance size and actual values are provided in Section 3.2. During the post-optimization phase, the cost function $\bar{C}(\hat{\mathcal{S}}_{best})$ is used to focus on reducing time-related infeasibilities if any, and to ensure that a feasible solution remains feasible after each move of the post-optimization phase. The VND sequentially explores relocate and exchange neighborhoods using the multi-trip improvements operators of Section 2.1. The relocate neighborhood relocates a chain of vertices from its current position to another one. The exchange neighborhood swaps two chains of vertices. Both neighborhoods consider moves within the same vehicle and moves implying two vehicles. Neighborhoods are tested in the following order: relocate-1, relocate-2, relocate-3, exchange-1, exchange-2, exchange-3, numbered according to the considered chain length. Whenever an improvement is found, the search starts again with the relocate-1 neighborhood and it stops if exchange-3 fails at finding an improvement. Consecutive depots are only deleted at the end of the post-optimization phase to avoid decreasing the number of trips too early.

Summary of the ALNSM Heuristic. The ALNSM method is summarized below as it was implemented for the configuration phase.

2.4 Adaptive Large Neighborhood Search Combined with Bin Packing (ALNSP)

In this section, we describe an ALNSP algorithm for MTVRPTW based on the routing-packing approach. The results yielded by this algorithm will be numerically compared with those obtained by the ALNSM in Section 3.2. The ALNSP algorithm iteratively destroys and recreates a relaxed VRPTW solution $\hat{\mathcal{X}}$, called a r-VRPTW solution, which is a set of trips that serve all the customers and respect the capacity constraints but may violate time windows and/or contain overtime. The size of this set of trips is not limited. The cost of a r-VRPTW solution is defined as $C(\hat{\mathcal{X}}) = D(\hat{\mathcal{X}}) + \alpha(TW(\hat{\mathcal{X}}) + OT(\hat{\mathcal{X}}))$ where $D(\hat{\mathcal{X}})$ is the total duration of all trips in $\hat{\mathcal{X}}$, $TW(\hat{\mathcal{X}})$ is the total time wrap usage of $\hat{\mathcal{X}}$, and $OT(\hat{\mathcal{X}})$ is the total overtime.

The removal and insertion heuristics do not employ any multi-trip operator, since trips are considered separately. To create r-MTVRPTW solutions, an assignment procedure allocates the trips of r-VRPTW

Algorithm 1 ALNSM

Construct an initial r-MTVRPTW solution $\hat{\mathcal{S}}$ using multi-trip operators
$\hat{\mathcal{S}}_{best} \leftarrow \hat{\mathcal{S}}$
$q = 1$; initialize the roulette wheel; initialize α
while the maximum run time is not reached **do**
    Roulette wheel: select a removal heuristic h_{rem} and an insertion heuristic h_{ins}
    Remove q customers from $\hat{\mathcal{S}}$ using h_{rem}, creating a partial relaxed solution
    Insert customers into the partial solution using h_{ins}, creating a solution $\hat{\mathcal{S}}'$
    if the acceptance criterion is met (cost measure: $\hat{C}$) **then**
        $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}}'$
        $q \leftarrow 1$
        Update the value of the adaptive penalty α (cost measure: $\bar{C}$)
    else
        $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q + 1 > q_{max}$
    end if
    if $\hat{\mathcal{S}}'$ is a new best solution **then**
        $\hat{\mathcal{S}}_{best} \leftarrow \hat{\mathcal{S}}'$
    end if
    Update the roulette wheel
end while
Apply post-optimization on $\hat{\mathcal{S}}_{best}$

solutions to the m available vehicles, as explained below.

Initial Solution. An initial r-VRPTW solution $\hat{\mathcal{X}}$ is built with an unbounded number of trips allowed. Namely, there is always exactly one empty trip available to insert customers. Unrouted customers are iteratively inserted at their best possible position using only the insertion scheme 1. At each iteration of this greedy procedure, the customer to be added is chosen so as to minimize the cost increment (with $\alpha = \alpha_{min}$) of the partial solution while preserving feasibility with respect to capacity constraints.

Destroy-and-Repair. The destroy-and-repair mechanism of the ALNSP is nearly the same as the one of the ALNSM except that it iteratively modifies r-VRPTW solutions instead of working on r-MTVRPTW solutions. The penalized objective function described above is used to explore the r-VRPTW search space. As it is the case during the creation of the initial solution, there is always exactly one empty trip available for the insertion of customers.

Solution Acceptance. Let $\psi \geq 1$ be a parameter called pre-acceptance factor. If the duration $D(\hat{\mathcal{X}}')$ of the candidate solution is not excessively deteriorated compared to that of the incumbent $D(\hat{\mathcal{X}})$, i.e., if $D(\hat{\mathcal{X}}') \leq \psi D(\hat{\mathcal{X}})$, an assignment procedure creates a r-MTVRPTW solution $\hat{\mathcal{S}}'$ by assigning the trips of $\hat{\mathcal{X}}'$ to the m available vehicles, and $D(\hat{\mathcal{S}}')$, $TW(\hat{\mathcal{S}}')$ and $OT(\hat{\mathcal{S}}')$ can be calculated. On the contrary, if $D(\hat{\mathcal{X}}') > \psi D(\hat{\mathcal{X}})$, then $\hat{\mathcal{X}}'$ is automatically discarded without going through the assignment phase.

Once a candidate r-MTVRPTW solution $\hat{\mathcal{S}}'$ is created, it is tested for acceptance with the exact same

procedure as in the ALNSM, i.e., $C(\hat{\mathcal{S}}')$ is compared to $C(\hat{\mathcal{S}})$, where $\hat{\mathcal{S}}$ is the r-MTVRPTW solution associated with the incumbent r-VRPTW solution $\hat{\mathcal{X}}$, using a simulated annealing framework.

In order to detect improving or new best solutions to accordingly update the roulette wheel, the comparison of r-MTVRPTW solutions is performed exactly as in the ALNSM, considering only those solutions that are pre-accepted.

Assignment. In the context of the MTVRP, the assignment phase can be viewed as a bin packing procedure where trips are items to be packed into bins representing the available vehicles. Durations of the trips are the item sizes and the maximum allowed duration for each vehicle is the bin size. In the MTVRPTW, however, the analogy is more complicated to establish since the duration of a trip, i.e., the corresponding item size, depends on its start time. Also, a feasible trip may become infeasible if its start time is shifted forward in time. As highlighted in Cattaruzza et al. (2016b), the assignment phase of an MTVRPTW can be viewed as a scheduling problem. Each vehicle is a machine, which in this work is continuously available for a limited duration, and each trip is a job with a time-dependent processing-time. Note that we kept the name "ALNSP" for the sake of consistency with the previous work of François et al. (2016) even if the assignment phase resembles more a scheduling problem than a packing problem.

In the assignment phase, the trips of a r-VRPTW solution $\hat{\mathcal{X}}$ are assigned to m available vehicles to create a r-MTVRPTW solution $\hat{\mathcal{S}}$. For that purpose, we developed a heuristic procedure that consists of a construction phase followed by an improvement phase. First, an initial assignment is created by filling one vehicle at a time as explained in Algorithm 2. Each trip with a duration exceeding D^{max} is assigned to an empty vehicle. Remaining empty vehicles are then filled one at a time, trying to keep the total infeasibility (time wrap and overtime) as low as possible. If one or more trips remain unassigned, they are added to the initial assignment through a greedy procedure. Then, the initial assignment is refined by means of a first-improvement descent procedure. Moves consist in: *1)* reassigning a trip to another vehicle, *2)* exchanging two trips between two vehicles. Improvement is measured by the decrease of $\bar{C}(\hat{\mathcal{S}}')$.

Post-Optimization. Once the maximum run time is reached, a VND tentatively improves the best r-MTVRPTW solution found. Neighborhoods are the same as in the ALNSM, except that chains to be relocated or exchanged contain customers of a single trip. After the VND terminates, the obtained solution is temporarily split into its constituting trips and the assignment procedure is called to see if the total duration can be improved by modifying the assignment or if feasibility can be achieved in the case of an infeasible solution.

Summary of the ALNSP Heuristic. The ALNSP algorithm is summarized in Algorithm 3.

Algorithm 2 Initial assignment

Let $\mathcal{L}_1$ be the list of trips whose minimum duration exceeds D^{max}.
while $\mathcal{L}_1 \neq \emptyset$ **do**
    Assign an element r of $\mathcal{L}_1$ to an empty vehicle k.
    Set $\tau_k = \tau_r$.
    Remove r from $\mathcal{L}_1$.
end while
Let $\mathcal{L}_2$ be the list of unassigned trips r sorted in increasing order of $E(\tau_r)$.
for each empty vehicle k **do**
    Set $\mathcal{L}_3 = \mathcal{L}_2$.
    while $\mathcal{L}_3 \neq \emptyset$ **do**
        Choose r as the element in $\mathcal{L}_3$ that minimizes $\Delta TW(\tau_k \oplus \tau_r) + \Delta OT(\tau_k \oplus \tau_r)$, break ties by choosing the element with the smallest $E(\tau_r)$.
        Set $\tau_k = \tau_k \oplus \tau_r$.
        Remove r from $\mathcal{L}_2$ and from $\mathcal{L}_3$.
        Remove from $\mathcal{L}_3$ every trip r' with $L(\tau_{r'}) < E(\tau_k) + D(\tau_k)$.
        Remove from $\mathcal{L}_3$ every trip r' with $D(\tau_k \oplus \tau_{r'}) > D^{max}$.
    end while
end for
while unassigned trips remain **do**
    Assign a trip r to a vehicle k so as to minimize the infeasibility increment $\Delta TW(\tau_k \oplus \tau_r) + \Delta OT(\tau_k \oplus \tau_r)$.
end while

Algorithm 3 ALNSP

Construct an initial r-VRPTW solution $\hat{\mathcal{X}}$
Apply the assignment procedure on $\hat{\mathcal{X}}$ to produce $\hat{\mathcal{S}}_{best}$
$q = 1$; initialize the roulette wheel; initialize α
while the maximum run rime is not reached **do**
    Roulette wheel: select a removal heuristic h_{rem} and an insertion heuristic h_{ins}
    Remove customers from $\hat{\mathcal{X}}$ using h_{rem}, creating a partial relaxed solution
    Insert customers into the partial solution using h_{ins}, creating a solution $\hat{\mathcal{X}}'$
    if the pre-acceptance criterion is met **then**
        Apply the assignment procedure on $\hat{\mathcal{X}}'$ to produce $\hat{\mathcal{S}}'$
        if the acceptance criterion is met **then**
            $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}}'$
            $q \leftarrow 1$
            Update the value of the adaptive penalty α
        else
            $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q + 1 > q_{max}$
        end if
        if $\hat{\mathcal{S}}'$ is a new best solution **then**
            $\hat{\mathcal{S}}_{best} \leftarrow \hat{\mathcal{S}}'$
        end if
    end if
    Update the roulette wheel
end while
Apply post-optimization on $\hat{\mathcal{S}}_{best}$

2.5 Removal and Insertion Heuristics

As mentioned in the introduction of Section 2, the ALNSM and the ALNSP algorithms use insertion and removal heuristics at each iteration in order to destroy and repair solutions. The ALNSM modifies r-MTVRPTW solutions employing the operators described in Section 2.1. The ALNSP modifies r-VRPTW solutions with classical removal and insertion of customers between two consecutive vertices of a trip. In both cases, removal heuristics define the criteria used to select customers to be removed, and insertion heuristics define the sequence and the positions of customer reinsertions.

2.5.1 Removal Heuristics.

The set of removal heuristics H_{rem} available for selection by the roulette wheel in the ALNSM or ALNSP is determined in the configuration phase. When a heuristic has different versions, each version can be integrated in the final implementation of the algorithms separately, regardless of the decision taken for other versions.

Random Removal. In the random removal heuristic, q customers are randomly removed from a solution $\hat{\mathcal{S}}$ or $\hat{\mathcal{X}}$ using a discrete uniform probability distribution.

Worst Removal. As proposed by Ropke and Pisinger (2006a), the worst removal heuristic tends to remove customers with the aim of obtaining important savings. Let i be a customer of a r-MTVRPTW solution $\hat{\mathcal{S}}$ (resp. a r-VRPTW solution $\hat{\mathcal{X}}$). We propose three measures of the savings, giving rise to three different versions of the heuristic *(a)*, *(b)*, and *(c)*. Let the cost of vehicle k serving customer i be $\tilde{C}_k = D_k + \alpha_w(TW_k + OT_k)$, where α_w is equal to *(a)* 0; *(b)* the penalty of the current iteration α, *(c)* a very large number. Similarly, let $\tilde{C}_k^{i^-}$ be the cost of vehicle k if customer i is removed, using the same penalty α_w. In all three versions, the savings associated to a customer i are computed as $\tilde{C}_k - \tilde{C}_k^{i^-}$. A ranked list of routed customers, denoted by $\mathcal{V}$, is obtained by sorting the customers in decreasing order of the savings. Then, q customers are selected out of $\mathcal{V}$ for removal. As proposed in Ropke and Pisinger (2006a), randomization is applied in a way that favors the selection of customers with smaller ranks in $\mathcal{V}$. This process is detailed in Algorithm 4.

Algorithm 4 Selection of q customers to be removed

1: Create the ranked list $\mathcal{V}$.
2: **while** the number of customers selected for removal is strictly smaller than q **do**
3: Set the value of $\nu \sim \mathcal{U}[0, 1[$.
4: Select customer i, whose index in $\mathcal{V}$ is $\lfloor \nu^{y_{rem}} \times |\mathcal{V}| \rfloor$, where $y_{rem} \geq 1$ is a parameter.
5: Remove customer i from $\mathcal{V}$.
6: **end while**

Once q customers have been selected, they are removed from the solution, which is updated accordingly. The parameter y_{rem}, called randomization factor, influences the probability of choosing elements

occupying the first positions of $\mathcal{V}$.

Shaw Removal. Shaw (1998) proposed the idea of removing customers based on their resemblance. In this heuristic, a customer i is first removed randomly. Then, Algorithm 4 is applied to remove $q-1$ customers, except that the sorting criterion of the ranked list $\mathcal{V}$ changes: remaining customers are now sorted in decreasing order of their resemblance measure with i. Let tt_{max} be the maximum travel time between any pair of customers, tw_{max} be the maximum absolute difference between the time window centers of any pair of customers, and d_{max} be the maximum demand. In this work, we define $1/p_{ij}$ as the resemblance measure of two customers i and j, where p_{ij} is equal to: *a)* $\Phi_{tt}(t_{ij}/tt_{max}) + \Phi_d(|d_i - d_j|/d_{max}) + \Phi_{tw}(\|(b_i - a_i)/2 - (b_j - a_j)/2\|/tw_{max})$, where Φ_{tt}, Φ_d, and Φ_{tw} are parameters, whose sum is equal to 1; *b)* (t_{ij}/tt_{max}); *c)* $(\|(b_i - a_i)/2 - (b_j - a_j)/2\|/tw_{max})$. Each measure gives rise to a different version of Shaw's removal heuristic.

Historical Removal. Ropke and Pisinger (2006b) proposed to remove customers based on historical information. The principle is to apply Shaw removal heuristic while incorporating historical information to compute the resemblance between two customers. As in François et al. (2016), we use $1/h_{ij}$ to measure the resemblance between customers i and j, where h_{ij} is the average cost of the λ_H best solutions found so far with i and j placed in the same trip. In the case of the ALNSM, we also use a second version of the heuristic where trip-related information is replaced with vehicle-related information. A dedicated memory stores the λ_H best solution values for each couple of customers. The costs of different solutions may only be compared if the penalty for time-related infeasibilities is kept constant. This is why the penalty of the cost function is set to α_H when computing costs to store in the dedicated memory. The respective values of α_H and λ_H are determined during the configuration phase.

Trip Removal. In the trip removal heuristic, trips are completely removed one by one, in increasing order of the number of customers they serve, until at least q customers have been removed. Ties are broken arbitrarily. This heuristic is especially important for the ALNSP since it allows a fast reduction of the number of trips in a candidate r-VRPTW solution.

2.5.2 Insertion Heuristics.

The insertion heuristics described below all follow the same principle: at each iteration, a customer is selected and inserted in a partial solution $\hat{\mathcal{S}}$ or $\hat{\mathcal{X}}$ until all customers have been routed.

At a given iteration of the ALNSM (resp. ALNSP) algorithm, to repair a solution $\hat{\mathcal{S}}$ (resp. $\hat{\mathcal{X}}$), the insertion heuristic picked by the roulette wheel iteratively selects one unrouted customer and inserts it between two consecutive vertices of a tour (resp. trip). For the ALNSM, as mentioned in Section 2.3, when evaluating the insertion of a customer at a given position of a tour, insertion scheme 4 only needs

to be evaluated if schemes 2 and 3 are infeasible with respect to capacity constraints; and schemes 2 and 3 need to be evaluated only if scheme 1 insertion violates those constraints. In the following, when evaluating the cost of inserting a customer at a given position of a tour, we always select the insertion scheme that yields the least cost increment. For the ALNSP, depots are never inserted: only scheme 1 is considered.

Heuristics implemented in this work differ in the criterion used to select the customers. Once a customer is selected, it is always inserted at its best possible position in the partial solution, i.e., the insertion position that yields the least cost increment. The cost function used in all heuristics described below is either C or $\bar{C}$. The choice of the cost function used by all insertion heuristics is performed during the configuration phase by means of a single boolean parameter. In Table 1, heuristics are listed and the criteria used for customer selection are detailed. Each line of the table corresponds to a different heuristic or group of heuristics, that is, a greedy heuristic and three groups of regret heuristics. The second column provides the selection criterion of i, the customer to insert, at each iteration of the concerned heuristic. For regret heuristics, the criteria rely on a parameter x, whose values are given in the third column. Each value of x gives rise to a specific version of the concerned heuristic. For example, there are four versions of the trip-based regret heuristic. Each of these versions is considered by the roulette wheel as a separate heuristic.

Table 1: Insertion heuristics

Heuristic	Customer selection	x	Algorithm
Greedy	$i = \text{argmin}_{j \in U}\{\Delta C^{Pos}_{j,1}\}$	-	ALNSM, ALNSP
Trip-based regret	$i = \text{argmax}_{j \in U} \sum_{y=2,\ldots,x}\{\Delta C^{Trip}_{j,y} - \Delta C^{Trip}_{j,1}\}$	$x = 2, 3, 4, 5$	ALNSM, ALNSP
Vehicle-based regret	$i = \text{argmax}_{j \in U} \sum_{y=2,\ldots,x}\{\Delta C^{Veh}_{j,y} - \Delta C^{Veh}_{j,1}\}$	$x = 2, ..., \min\{4, m-1\}$	ALNSM
Position-based regret	$i = \text{argmax}_{j \in U} \sum_{y=2,\ldots,x}\{\Delta C^{Pos}_{j,y} - \Delta C^{Pos}_{j,1}\}$	$x = 2, 3, \lceil q/2 \rceil, q$	ALNSM, ALNSP

In Table 1, $\Delta C^{Pos}_{j,y}$ is the variation in the value of the cost function if customer j is inserted in the position with the y^{th} lowest insertion cost. $\Delta C^{Trip}_{j,y}$ (resp. $\Delta C^{Veh}_{j,y}$) is the variation in the value of the cost function if customer j is inserted in the best possible position in the trip (resp. vehicle) where its insertion cost is the y^{th} lowest. While the greedy heuristic iteratively inserts a customer that minimizes the cost increment at its best possible position, the customer selection criteria of regret heuristic incorporate information about different possible insertion positions. The resulting selection process prioritizes customers that may be more difficult to insert with a small cost increment later on.

The set of insertion heuristics H_{ins} available for selection by the roulette wheel in the ALNSM or ALNSP is determined in the configuration phase, each group of regret heuristics being considered as a whole.

3 Numerical Results

This section is organized as follows. First, we explain how the methods were configured, i.e, how the values of algorithmic parameters were determined. Next, to assess the quality of our methods, we confront them with the existing literature. Then, we compare both methods on new benchmark instances. Finally, we discuss the relevance of two objective functions, i.e., total travel time and the total duration.

3.1 Configuration of the Algorithms

Both the ALNSM and ALNSP algorithms are configured with the automatic configuration tool `irace` (López-Ibáñez et al., 2016). To configure an algorithm, `irace` uses three main inputs: *1)* the algorithm itself, *2)* a set of training instances, *3)* the list of parameters to configure along with their respective types and ranges. The user also defines the total number of runs to be performed, called tuning budget. In this context, a run corresponds to the execution of the tested algorithm with a given parameter configuration and a given random seed on a given training instance. Also, since we consider instances that might be hard to solve or infeasible, we provide `irace` with a modified objective function, that includes a very big penalization factor for time-related infeasibilities, used in order to compare parameter configurations. At the end of an iterative racing process described in López-Ibáñez et al. (2016), `irace` returns a set of suitable configurations, said to be "elite", each consisting of a set of values assigned to algorithmic parameters. Amongst these elite configurations, for each algorithm, we choose the one recommended by `irace`, that we call "final configuration", and that gives rise to the ALNSM and ALNSP implementations used in the numerical experiments.

We configure both algorithms twice, to find suitable parameters values with the aim of minimizing either the total duration, or the total travel time. In the latter case, in order to guide the search properly, the total traveling time replaces the total duration in the definition of the cost measures $\hat{C}$ and $\bar{C}$ described in Section 2.2. In the following, we denote by ALNSM_T and ALNSM_D, ALNSM final configurations aiming to minimize the travel time and the duration respectively. Similarly, final ALNSP configurations are called ALNSP_T and ALNSP_D. In Section 3.3, configurations ALNSM_D and ALNSP_D are compared on our new benchmark instances with the objective of minimizing the total duration. On the contrary, the comparison of our methods with those of previous authors that minimize the total travel time rely on configurations ALNSM_T and ALNSP_T (See Section 3.2).

The twelve training instances provided to `irace` and the new benchmark instances described in Section 3.3 are two disjoint sets created with the same instance generator. By using disjoint sets, we avoid biasing the results by tuning the algorithm on a subset of the benchmark instances. Even if this is a well-known good practice, it is not broadly applied within the vehicle routing community.

The training budget for each algorithm is 50 000 runs, with a maximum run time for each run set to 250 seconds. The training phase is performed on a cluster with 128 compute nodes, each having two

8-cores Intel E5-2650 processors at 2.0 GHz and 64 GB of RAM (4 GB/core). The process is parallelized (up to 160 cores), and each configuration takes up to 48 hours to be performed.

The configuration tool determines not only the value of numerical parameters, but also the heuristics to be included in H_{rem} and H_{ins} for the final design of the algorithms as mentioned in Section 2.5. In other words, we implement a set of heuristics and let the configuration tool decide which ones are suitable depending on the method and objective function considered. Each final configuration is in fact a contextualized implementation of the ALNSM or ALNSP method, with appropriate algorithmic design choices. François et al. (2016) showed that an a priori heuristic selection is crucial and that the roulette wheel mechanism cannot effectively compensate for the lack of it. This indicates that eliminating elements of H_{rem} and H_{ins} offline, before the online selection through the roulette wheel, is crucial. In Table 2, the list of heuristics included in each final configuration is given: a *TRUE* (T) value for a given heuristic in the column corresponding to a final configuration indicates that this heuristic is included in this configuration; a *FALSE* (F) value indicates that it is excluded.

Table 2: Heuristics in ALNSM and ALNSP configurations

	ALNSM		ALNSP	
Heuristics	ALNSM$_D$	ALNSM$_T$	ALNSP$_D$	ALNSP$_T$
Removal heuristics				
Use random removal.	F	T	F	T
Use worst removal - version *(a)*.	T	F	T	T
Use worst removal - version *(b)*.	T	T	T	F
Use worst removal - version *(c)*.	F	T	F	F
Use Shaw removal - version *(a)*.	F	F	F	F
Use Shaw removal - version *(b)*.	T	F	T	T
Use Shaw removal - version *(c)*.	T	T	T	F
Use historical removal with trip-based information.	F	F	F	F
Use historical removal with vehicle-based information.	F	F	-	-
Use trip removal.	F	T	F	T
Design option: include removal heuristics without the merge operator.	T	T	-	-
Insertion heuristics				
Use greedy insertion	T	T	T	T
Use position-based regret insertion.	T	T	F	F
Use trip-based regret insertion.	T	F	T	T
Use vehicle-based regret insertion.	T	F	-	-
Design option: use insertion costs with a huge penalty	F	F	F	F

Note from Table 2 that each removal heuristic is rejected by at least one of the four final configurations. This shows the importance of the configuration tool to select the adequate algorithm components depending of the context.

Historical removal heuristics and version *(a)* of Shaw removal are never selected. When analyzing the results produced by `irace`, we observe that none of the elite configurations uses these three heuristics while the results may vary for the others.

In Table 3, the type and range of numerical parameters (either real or integer) are specified, as well as their values in the four final configurations. Some of those parameters must be configured or not depending on the value taken by some others. For example, the weights used in version *(a)* of Shaw removal heuristic need to be configured only if the parameter corresponding to the use of this heuristic is *TRUE*. Unless otherwise specified, the configuration of real parameters is performed with a precision of two digits.

Table 3: Numerical parameter values in ALNSM and ALNSP configurations

				ALNSM		ALNSP	
	Name	Type	Authorized range	$ALNSM_D$	$ALNSM_T$	$ALNSP_D$	$ALNSP_T$
Objective function (2.2)	α_{min}	integer	[1,100]	5	48	46	72
	α_{max}	integer, step 10	[α_{min},1000]	773	48	210	528
	μ	real	[1,2]	1.45	-	1.24	1.56
	ξ	integer, step 10	[0,100]	10	-	30	10
Simulated annealing (2.3: *Solution acceptance*)	θ_0	real, 3 digits	[0.01,0.1]	0.015	0.077	0.041	0.016
	κ	real	[0,1]	0.24	0.27	0.17	0.85
	η	real, 3 digits	[0.9,1]	0.949	0.973	0.988	0.961
Roulette wheel (2.3: *Destroy-and-repair*)	σ_1	real	[0,1]	0.18	0.11	0.31	0.08
	σ_2	real	[0,1 − σ_1]	0.62	0.51	0.33	0.49
	ρ	real	[0,1]	0.30	0.39	0.64	0.25
	Θ	integer, step 100	[500,5000]	3600	2700	1900	2500
Bounds on q (2.3: *Destroy-and-repair*)	υ	real	[0.05,0.40]	0.17	0.18	0.21	0.29
	δ	integer	[5,15]	14	6	8	13
Randomization factor (2.5.1)	p	real	[1.10,10]	8.05	2.13	7.29	1.71
Shaw removal (2.5.1: *Shaw removal*)	Φ_{tt}	real	[0,1]	-	-	-	-
	Φ_{tw}	real	[0,1]	-	-	-	-
Historical removal (2.5.1 : *Historical removal*)	λ_H	integer, step 5	[5,100]	-	-	-	-
	α_H	integer	[0,1000]	-	-	-	-
ALNSP pre-acceptance factor (2.3: *Solution acceptance*)	ψ	real, 3 digits	[1,1.10]	-	-	1.078	1.069

3.2 Comparison with other Authors

We compare the results of our algorithms with those of the memetic algorithm of Cattaruzza et al. (2016a) and with the optimal solutions provided by the exact methods of Hernandez et al. (2016). For this purpose, we modify our objective function and minimize the total travel time instead of the total duration as explained in Section 3.1. The total travel time replaces the total duration in all the cost functions used to guide the search, including the criteria and cost functions of insertion and removal heuristics. Doing so, and setting the maximum allowed duration for each vehicle to the size of the planning horizon, we are able to provide a meaningful comparison with Cattaruzza et al. (2016a) on two sets of instances: those of Hernandez et al. (2016) and those of Cattaruzza et al. (2016a).

Hernandez et al. (2016) modify Solomon's instances of groups C2, R2, and RC2, i.e., instances with large planning horizon. Travel times are calculated as Euclidean distances between each pair of

vertices, truncated to the first decimal place. Vehicle capacities are fixed to 100. The loading factor γ is equal to 0.2. Instances with 25 customers are generated by considering only the 25 first customers of Solomon's graphs and 2 vehicles. Hernandez et al. (2016) provide optimal solutions for 25 out of those 27 instances with 25 customers. Hernandez et al. (2013) also report optimal results for 5 out of 27 instances similarly created with 50 customers and 4 vehicles. Cattaruzza et al. (2016a) extend this instance set by considering 27 instances, generated following the same methodology, with 100 customers and 8 vehicles. In the following, these instances are denoted by HFGN-25, HFGN-50, and HFGN-100, depending on the number of customers.

Cattaruzza et al. (2016a) also modify Solomon's instances by introducing release dates and modifying the number of vehicles and their capacities considering travel times as Euclidean distances without rounding nor truncation. There is no loading time ($\gamma = 0$) and the vehicle capacity is set to half its original value. The fleet size is reported in Table 7, where we provide a comparison on the subset of their instances in which all release dates are equal to 0. In the following, these instances are denoted by CAF-100.

Thus, in total, we have 81 instances denoted HFGN-25, HFGN-50, and HFGN-100, involving respectively 25, 50, and 100 customers, and 56 instances denoted CAF-100 involving 100 customers.

The maximum run time is set to a time limit of 20, 50, and 250 seconds for instances with 25, 50, and 100 customers respectively for both configurations $ALNSM_T$ and $ALNSP_T$. Tests are performed on an Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz. Cattaruzza et al. (2016a) use an Intel(R) Xeon(R) W3550 CPU @ 3.07GHz. Their maximum run time is set to 60 seconds for instances with 25 and 50 customers, and 300 seconds for instances with 100 customers.

As shown in Table 4, both ALNSM and ALNSP produce the same solution value over five runs in all cases. This value is known to be optimal for 25 out of 27 instances. For the two instances that are not yet solved to optimality, the best known solution of Cattaruzza et al. (2016a) is reached.

In Tables 5, 6, and 7, results of the ALNSM and ALNSP algorithms are compared with those of Cattaruzza et al. (2016a) on instance sets HFGN-50, HFGN-100, and CAF-100 respectively. For Table 5, the optimal value of Hernandez et al. (2013) is reported when known. The last row of each table contains a summarized comparison: the average percentage gap (over all instances) between the best ALNSM or ALNSP solution values and the best solution value of Cattaruzza et al. (2016a) is given, as well as the average percentage gap between the average ALNSM or ALNSP solution values over five runs and the average solution value of Cattaruzza et al. (2016a). Currently best known values are in bold. Both methods outperform the memetic algorithm of Cattaruzza et al. (2016a), with the ALNSM providing on average the best improvement.

Note that, for both algorithms, the post-optimization procedure only yields tiny improvements of the objective function value for a few instances. However, since it is extremely fast (less than one second for

Table 4: Results on HFGN-25 instances ($n = 25, m = 2, \gamma = 0.2$)

			Number of best known		
VRP inst.	opt	best known	CAF2016	ALNSM	ALNSP
C201	380.8		4	5	5
C202	368.6		5	5	5
C203	361.7		5	5	5
C204	358.8		5	5	5
C205	377.2		5	5	5
C206	367.2		5	5	5
C207	359.1		5	5	5
C208	360.9		5	5	5
R201	554.6		5	5	5
R202	485.0		5	5	5
R203	444.2		5	5	5
R204	407.5		5	5	5
R205	448.4		5	5	5
R206	413.9		5	5	5
R207	400.1		5	5	5
R208	394.3		5	5	5
R209	418.3		5	5	5
R210	448.3		5	5	5
R211	400.1		5	5	5
RC201	660.0		5	5	5
RC202	596.8		5	5	5
RC203	530.1		5	5	5
RC204		518.0	5	5	5
RC205	605.3		5	5	5
RC206	575.1		4	5	5
RC207	528.2		5	5	5
RC208		506.4	5	5	5

instances with 100 customers), we keep it included in the final design of our algorithms.

Table 5: Results on HFGN-50 instances ($n = 50, m = 4, \gamma = 0.2$)

	HFGN2013	CAF2016		ALNSM		ALNSP	
VRP inst.	opt	best cost	average cost	best cost	average cost	best cost	average cost
C201		**714.2**	714.20	**714.2**	714.20	**714.2**	714.20
C202		**700.1**	700.38	**700.1**	700.10	**700.1**	700.64
C203		**688.0**	689.34	**688.0**	688.00	**688.0**	688.82
C204		**685.1**	685.10	**685.1**	685.10	**685.1**	685.56
C205		700.0	703.52	**699.1**	699.84	**699.1**	701.22
C206		**694.6**	696.92	**694.6**	695.48	**694.6**	695.68
C207		689.7	690.38	**688.6**	688.60	**688.6**	688.84
C208		**688.6**	688.60	**688.6**	688.60	**688.6**	688.60
R201	909.8	**909.8**	917.08	**909.8**	914.36	**909.8**	916.80
R202	816.0	**816.0**	816.00	**816.0**	816.68	816.3	817.80
R203		**742.4**	743.40	**742.4**	742.40	**742.4**	743.80
R204		**702.3**	704.38	**702.3**	702.30	**702.3**	702.30
R205	807.3	**807.3**	808.74	808.4	808.96	**807.3**	809.86
R206		**758.2**	760.96	**758.2**	758.56	**758.2**	758.38
R207		**715.7**	715.70	**715.7**	715.70	**715.7**	715.70
R208		**699.6**	700.60	**699.6**	699.60	**699.6**	699.60
R209		746.0	746.00	**745.0**	745.80	746.0	746.54
R210		**777.2**	779.22	**777.2**	778.76	**777.2**	779.54
R211		717.4	722.02	**716.8**	717.44	**716.8**	719.26
RC201	1096.6	**1096.6**	1096.60	1097.6	1097.60	1097.6	1098.82
RC202	1001.6	1038.6	1038.60	**1027.0**	1036.28	1038.6	1038.92
RC203		**941.2**	941.20	**941.2**	941.20	**941.2**	941.20
RC204		**915.9**	915.90	**915.9**	915.90	**915.9**	915.90
RC205		**1058.7**	1058.70	**1058.7**	1058.70	**1058.7**	1058.70
RC206		**1027.4**	1032.12	**1027.4**	1031.96	**1027.4**	1030.76
RC207		**941.7**	941.70	**941.7**	941.70	**941.7**	941.70
RC208		**916.8**	916.80	**916.8**	916.80	**916.8**	916.80
Gap compared to CAF2016 best and average				-0.05%	-0.12%	-0.01%	-0.05%

Table 6: Results on HFGN-100 instances ($n = 100, m = 8, \gamma = 0.2$)

	CAF2016		ALNSM		ALNSP	
VRP inst.	best cost	average cost	best cost	average cost	best cost	average cost
C201	1488.9	1500.22	**1486.0**	1486.72	1487.2	1489.50
C202	1479.3	1486.94	**1473.9**	1479.78	1482.3	1485.58
C203	1467.3	1471.20	**1466.8**	1467.52	1467.7	1470.00
C204	**1453.6**	1455.46	1454.2	1457.24	1454.4	1460.54
C205	1477.1	1483.04	**1474.3**	1481.66	1478.9	1484.94
C206	1464.7	1473.38	1463.2	1464.70	**1462.7**	1464.28
C207	**1464.2**	1470.36	1465.2	1468.32	1468.0	1470.00
C208	**1459.4**	1465.86	1461.1	1463.10	1461.8	1464.20
R201	1449.7	1464.32	**1412.4**	1419.30	1424.0	1426.28
R202	1343.3	1352.70	**1314.8**	1320.98	1323.1	1326.52
R203	1222.2	1232.50	1213.2	1219.56	**1211.2**	1219.78
R204	1165.6	1172.54	**1163.2**	1167.88	1170.3	1173.60
R205	1292.2	1315.08	**1281.8**	1289.98	1295.1	1302.72
R206	1239.9	1249.76	**1226.5**	1231.58	1234.6	1237.22
R207	1194.3	1200.76	**1189.7**	1197.92	1192.6	1196.60
R208	1159.8	1164.56	**1159.5**	1162.36	1162.2	1164.14
R209	1234.5	1248.42	**1225.8**	1228.34	1235.8	1240.26
R210	1247.5	1253.24	**1222.4**	1233.20	1233.9	1238.48
R211	**1170.5**	1182.38	1172.9	1177.76	1179.3	1182.34
RC201	1843.6	1862.40	1826.8	1842.78	**1820.9**	1835.62
RC202	1733.9	1740.34	**1703.6**	1711.98	1719.2	1724.00
RC203	**1618.6**	1624.24	1619.8	1624.52	1623.2	1628.04
RC204	**1579.1**	1581.36	1580.6	1582.58	1579.5	1582.04
RC205	1759.8	1776.68	**1758.3**	1760.14	1760.2	1767.90
RC206	1731.1	1747.24	**1718.2**	1725.18	1726.7	1729.82
RC207	1656.7	1662.96	**1656.3**	1665.16	1664.0	1671.02
RC208	**1580.9**	1585.44	1582.8	1585.30	1586.1	1588.96
Gap compared to CAF2016 best and average			-0.53%	-0.79%	-0.18%	-0.51%

Table 7: Results on CAF-100 instances ($n = 100, \gamma = 0$)

		CAF2016		ALNSM		ALNSP	
VRP inst.	m	best cost	average cost	best cost	average cost	best cost	average cost
C201	3	**777.48**	777.48	778.62	778.62	778.62	778.62
C202	4	**718.69**	724.58	**718.69**	725.21	724.20	728.23
C203	5	700.20	711.06	**699.87**	699.87	709.80	709.91
C204	5	695.12	698.17	**693.63**	693.63	707.14	709.57
C205	3	**767.55**	770.21	770.67	770.67	770.67	776.85
C206	3	**747.14**	750.42	**747.14**	747.14	**747.14**	752.52
C207	3	**746.62**	748.66	**746.62**	746.62	**746.62**	746.62
C208	3	**741.58**	742.09	**741.58**	741.59	758.52	758.52
R201	4	1272.47	1287.21	**1254.69**	1256.45	1255.15	1255.92
R202	3	1272.72	1278.20	**1226.18**	1254.77	1316.02	1335.09
R203	3	966.35	976.30	971.56	972.52	**959.82**	992.51
R204	3	779.22	787.31	**760.20**	772.32	766.08	777.00
R205	3	1047.75	1089.24	**1018.39**	1030.91	1031.14	1045.36
R206	3	944.58	962.93	**913.02**	917.95	916.58	938.98
R207	3	849.64	862.07	**819.36**	828.84	833.00	858.22
R208	3	735.49	738.52	**717.59**	718.32	731.20	737.06
R209	3	944.06	961.83	**918.48**	926.18	952.22	957.65
R210	3	985.66	1001.98	968.35	973.14	**967.20**	973.89
R211	4	772.99	779.80	757.85	759.91	**755.78**	762.23
RC201	4	1424.18	1459.26	**1413.52**	1434.29	1425.87	1450.94
RC202	4	1171.86	1196.66	**1168.57**	1184.78	1171.95	1206.18
RC203	3	1108.21	1150.94	**1107.66**	1113.82	1112.74	1127.43
RC204	4	806.44	809.59	**801.92**	803.69	812.59	814.89
RC205	4	1321.64	1354.18	**1314.94**	1322.30	1325.72	1332.93
RC206	3	1325.01	1385.87	**1205.29**	1226.49	1261.73	1294.20
RC207	4	1042.03	1050.24	**1015.68**	1028.50	**1015.68**	1030.33
RC208	4	803.59	818.23	790.03	792.25	**794.78**	798.73
C101	12	1529.66	1568.27	**1519.38**	1526.45	1526.12	1538.44
C102	10	1675.75	1715.14	**1583.05**	1621.44	1633.73	1668.62
C103	11	1452.69	1524.98	1448.05	1452.52	**1432.92**	1436.35
C104	13	1384.78	1407.10	1405.89	1412.20	**1381.45**	1383.59
C105	11	1550.02	1709.18	**1540.06**	1550.03	1547.20	1555.87
C106	11	1592.13	1624.38	**1546.25**	1561.64	1570.30	1591.29
C107	11	1513.19	1527.41	**1471.73**	1483.53	1482.07	1484.99
C108	10	1545.94	1659.66	**1482.18**	1510.23	1516.02	1554.62
C109	10	1496.65	1538.70	1449.61	1454.66	**1441.00**	1457.28
R101	22	1671.77	1678.46	1660.18	1667.17	**1644.64**	1644.95
R102	19	1498.23	1502.58	1506.56	1508.58	**1485.71**	1485.71
R103	18	1288.44	1298.49	**1290.22**	1293.06	1291.83	1292.28
R104	17	1177.88	1190.59	1186.18	1189.08	**1177.88**	1180.45
R105	17	1421.19	1433.05	1433.65	1437.03	**1410.58**	1413.05
R106	15	1361.02	1365.95	**1328.65**	1329.83	1354.53	1357.81
R107	16	1235.15	1245.09	**1224.49**	1231.60	1245.14	1252.90
R108	16	1187.36	1193.28	**1176.40**	1179.75	1185.72	1188.41
R109	15	1307.25	1316.19	1307.04	1314.98	**1286.22**	1288.68
R110	15	1246.99	1253.13	1257.13	1262.61	**1226.96**	1231.51
R111	15	1236.23	1244.99	1235.76	1245.63	**1214.91**	1219.54
R112	18	1182.72	1191.47	**1174.00**	1175.77	1191.51	1194.75
RC101	19	1805.40	1828.30	**1799.15**	1799.63	1815.49	1823.73
RC102	17	1746.02	1759.63	1731.21	1747.26	**1722.07**	1741.14
RC103	18	1637.38	1641.92	1648.37	1660.55	**1634.15**	1634.29
RC104	19	1582.81	1583.46	1585.95	1588.16	**1582.81**	1583.05
RC105	18	1752.66	1759.14	**1732.47**	1734.72	1737.91	1750.42
RC106	16	1750.52	1764.37	**1699.70**	1715.15	1747.95	1751.87
RC107	18	1615.05	1618.37	**1613.48**	1613.60	1632.25	1636.25
RC108	18	1581.78	1587.03	1588.51	1592.14	**1580.88**	1582.00
Gap compared to CAF2016's best and average				-1.17%	-2.05%	-0.58%	-1.25%

3.3 Results for New MTVRPTW Instances

Creating suitable instances, where multiple trips naturally arise, by modifying Solomon's instances is not a straightforward task, especially when the maximum allowed duration per vehicle is shorter than the planning horizon. Indeed, multiple trips are favored by decreasing the vehicle capacities, implying an increase in the number of vehicles to be used. The fleet size has to be increased even more when the allowed duration for each vehicle is shortened, but this decreases the necessity to create multiple trips. Let us consider CAF-100 instances of groups R1 and RC1. For these instance groups, Cattaruzza et al. (2016a) use a number of vehicles ranging from 15 to 22. For most instances of Table 7, the number of trips obtained in the solution is nearly the same as the number of vehicles. Logically, if the maximum allowed duration per vehicle is shortened, and the fleet size consequently increased for each instance, then each vehicle will serve very few customers, often even only one or two per trip, and the number of trips per vehicle will tend to decrease.

For these reasons, we create a new set of instances with 100 customers that we believe have suitable characteristics for MTVRPTW variants. We follow the general philosophy of Solomon by dividing our instances into three groups of geographical customer repartitions (clustered (C), random (R), and random clustered (RC)) and two planning horizon sizes (short (1) with size 600 and long (2) with size 1200), while the maximum allowed duration per vehicle D^{max} is 480 in all cases.

For each of the six resulting groups, we generate four different instance bases. Each basis is defined by the geographical coordinates of the customers and the depot, as well as the customer demands, time window centers, and service times. For each of the 24 obtained instance bases, three sizes of time windows are considered: large (L), with size 240, medium (M), with size 120, and small (S), with size 60. The combination of the 24 initial instance bases with three time window sizes results in 72 instance graphs. Each of these is coupled with three values of m to create the final set of instances. For all instances, the vehicle capacity is fixed at 100 and the loading factor is 0.2.

The names of the 72 instance graphs begin with the geographical type, followed by the size of the planning horizon and the time window size. For example, graph names beginning with R_1_M indicate a random geographical repartition of the customers, a small planning horizon, and medium time windows.

The values of m chosen for each one of the 72 graphs are respectively the minimum value of m for which we were able to obtain a feasible solution, v, along with $v + 1$ and $v + 2$. For example, if for a given instance graph, solution values are reported for $m \in \{7, 8, 9\}$, then it means that we could not find any feasible solution for $m = 6$ on the same instance graph. Five graphs that could not be solved using less than 15 vehicles have been discarded, since we try to mimic a realistic multi-trip context. This leads to a final set of 201 MTVRPTW instances.

The numerical parameters of the instances and the methods employed to generate customer clusters, time windows, and demands are detailed in the Appendix. The instance files, and the Julia notebook

used to create them, are available at: http://hdl.handle.net/2268/216743.

In the following, we report all the results based on the variable part of the objective function, i.e., the sum of the total travel time and the total waiting time, denoted as Z. Indeed, the total service and loading time is a constant, equal to 1200, for all instances.

Tables 8, 9, and 10 yield detailed results for our 201 new MTVRPTW instances. Since the objective is to minimize the total duration, configurations $ALNSM_D$ and $ALNSP_D$ are used. Five runs are performed on each instance with each algorithm. The maximum run time is set to 250 seconds. Columns *feas*, Z^* and $\bar{Z}$ respectively give the number of feasible solutions found, and the best and average solution value over those runs who produced a feasible solution. Two more columns report the waiting time as a percentage of Z^* and the number of trips for the best solution. Note from Table 8 that none of the algorithms produced a feasible solution for instance graph R_2_M_50 with $m = 9$. However, feasible solutions are provided by configuration $ALNSM_T$, which was run to provide results for the next section. The ALNSM and ALNSP algorithms respectively find feasible solutions for 200 and 194 out of the 201 new instances. Over 1050 runs, the ALNSM and ALNSP algorithms find a feasible solution 953 and 929 times respectively. The best run of the ALNSM algorithm outperforms the one of the ALNSP for 164 out of 200 instances. We notice that for the vast majority of instances, waiting times are less than 2% of Z^* and they logically tend to decrease as the size of the time windows increases.

Table 8: Results for R instances

Instance		ALNSM					ALNSP				
graph	m	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)
R_1_S_35	7	3	2080.0	2092.03	3.24	33	2	2085.1	2099.90	2.09	31
R_1_S_35	8	5	1834.9	1857.02	0.13	27	5	1866.7	1880.06	0.54	27
R_1_S_35	9	5	1796.9	1807.82	1.04	27	5	1800.9	1825.64	0.39	27
R_1_S_40	8	5	2236.7	2263.64	0.97	26	5	2272.9	2293.52	0.57	27
R_1_S_40	9	5	2197.3	2217.80	0.10	24	5	2202.8	2232.10	1.12	25
R_1_S_40	10	5	2184.4	2202.10	2.00	25	5	2171.5	2204.14	1.00	25
R_1_S_45	9	5	2515.4	2553.78	1.16	26	5	2568.5	2590.86	1.46	27
R_1_S_45	10	5	2438.7	2460.98	0.53	26	5	2426.8	2465.44	0.56	25
R_1_S_45	11	5	2434.8	2443.42	0.24	24	5	2452.9	2459.16	0.11	25
R_1_S_50	9	5	2789.6	2815.86	0.99	29	5	2836.1	2870.86	0.85	31
R_1_S_50	10	5	2768.9	2799.18	1.37	29	5	2789.3	2818.74	0.76	27
R_1_S_50	11	5	2744.5	2765.34	0.71	26	5	2752.6	2789.92	0.55	27
R_1_M_35	7	5	1640.7	1655.94	0.55	27	5	1638.2	1671.76	0.16	27
R_1_M_35	8	5	1625.9	1642.14	0.17	25	5	1629.6	1643.86	0.00	25
R_1_M_35	9	5	1625.0	1632.68	0.17	27	5	1630.6	1647.08	0.04	25
R_1_M_40	7	5	1956.9	2007.76	0.03	25	5	1964.9	1998.10	0.75	26
R_1_M_40	8	5	1950.9	1960.64	0.88	24	5	1972.3	1985.44	0.28	25
R_1_M_40	9	5	1940.4	1948.12	0.00	25	5	1950.1	1972.40	0.03	23
R_1_M_45	8	5	2264.9	2283.34	0.31	26	5	2270.3	2306.10	0.56	26
R_1_M_45	9	5	2245.1	2261.20	0.24	26	5	2279.3	2287.34	0.36	27
R_1_M_45	10	5	2231.8	2256.80	0.24	25	5	2241.3	2267.08	0.59	25
R_1_M_50	8	5	2458.0	2477.74	0.26	28	4	2474.9	2564.45	0.62	29
R_1_M_50	9	5	2444.7	2454.36	0.08	28	5	2458.2	2482.72	0.32	27
R_1_M_50	10	5	2414.4	2442.02	0.09	25	5	2447.9	2469.46	0.20	27
R_1_L_35	6	5	1508.3	1514.98	0.00	27	5	1505.8	1516.84	0.00	26
R_1_L_35	7	5	1504.5	1518.30	0.05	26	5	1499.6	1520.34	0.00	26
R_1_L_35	8	5	1503.6	1518.18	0.00	26	5	1523.9	1526.48	0.07	26
R_1_L_40	7	5	1770.7	1780.16	0.00	25	5	1774.1	1787.58	0.00	25
R_1_L_40	8	5	1775.5	1784.72	0.00	25	5	1765.0	1785.20	0.00	25
R_1_L_40	9	5	1766.6	1776.60	0.10	25	5	1766.7	1786.16	0.00	23
R_1_L_45	7	5	2042.2	2089.10	0.01	25	3	2110.8	2124.40	0.00	27
R_1_L_45	8	5	2031.9	2039.12	0.06	25	5	2031.6	2043.72	0.01	25
R_1_L_45	9	5	2030.5	2034.46	0.01	25	5	2030.5	2034.84	0.01	24
R_1_L_50	8	5	2274.1	2285.50	0.00	27	5	2287.1	2298.62	0.00	27
R_1_L_50	9	5	2274.0	2285.68	0.00	26	5	2273.2	2287.80	0.00	27
R_1_L_50	10	5	2269.8	2278.84	0.00	27	5	2282.7	2299.04	0.14	26
R_2_S_35	9	1	2600.9	2600.90	1.70	34	0				
R_2_S_35	10	5	2394.9	2413.96	1.54	30	5	2424.7	2444.50	1.76	31
R_2_S_35	11	5	2373.7	2399.18	0.98	30	5	2423.0	2428.30	2.33	31
R_2_S_40	10	5	2569.0	2576.20	1.12	25	5	2567.3	2587.92	1.57	26
R_2_S_40	11	5	2546.8	2553.98	1.32	23	5	2554.5	2566.26	2.55	25
R_2_S_40	12	5	2531.2	2538.04	2.14	26	5	2534.3	2549.18	2.54	22
R_2_S_45	11	5	3011.9	3052.14	1.40	32	5	3034.9	3091.02	0.76	31
R_2_S_45	12	5	2924.2	2933.00	0.90	30	5	2948.1	2970.20	0.61	30
R_2_S_45	13	5	2901.9	2914.08	1.06	28	5	2911.5	2936.04	0.80	28
R_2_S_50	11	5	3376.3	3382.62	2.68	29	5	3411.4	3417.20	0.64	28
R_2_S_50	12	5	3268.6	3302.72	2.67	27	5	3295.8	3320.82	2.46	27
R_2_S_50	13	5	3253.4	3262.84	3.55	25	5	3263.2	3289.78	4.00	26
R_2_M_35	8	5	2223.2	2249.44	0.07	32	5	2204.4	2284.26	0.82	30
R_2_M_35	9	5	2179.2	2192.06	1.14	31	5	2190.0	2204.68	0.00	30
R_2_M_35	10	5	2159.4	2168.72	0.66	29	5	2173.3	2185.96	0.58	29
R_2_M_40	8	5	2335.5	2365.68	0.51	24	5	2385.6	2401.82	1.26	25
R_2_M_40	9	5	2279.8	2290.10	0.72	24	5	2282.6	2299.38	0.20	23
R_2_M_40	10	5	2246.9	2257.70	0.01	21	5	2240.7	2266.02	0.06	20
R_2_M_45	9	4	2724.1	2769.88	0.80	29	5	2772.6	2855.00	1.06	31
R_2_M_45	10	5	2604.3	2662.28	0.00	29	5	2681.5	2688.06	0.57	30
R_2_M_45	11	5	2614.0	2618.92	0.00	28	5	2634.0	2652.24	0.68	29
R_2_M_50	9	0					0				
R_2_M_50	10	5	2843.0	2855.28	0.53	27	5	2855.9	2876.92	0.99	26
R_2_M_50	11	5	2803.4	2812.86	1.19	25	5	2822.6	2851.24	0.53	26
R_2_L_35	7	5	1957.7	1996.26	0.00	29	5	1974.0	2052.18	0.19	29
R_2_L_35	8	5	1947.6	1954.74	0.00	28	5	1946.2	1966.18	0.12	29
R_2_L_35	9	5	1941.1	1948.18	0.00	29	5	1949.5	1954.20	0.00	29
R_2_L_40	7	0					1	2102.8	2102.80	0.05	24
R_2_L_40	8	5	2009.0	2017.54	0.02	23	5	2030.7	2045.40	0.20	23
R_2_L_40	9	5	1981.7	1995.56	0.42	22	5	1993.5	2011.48	0.01	22
R_2_L_45	8	5	2366.9	2375.56	0.35	29	5	2373.3	2399.08	0.00	28
R_2_L_45	9	5	2350.0	2356.24	0.14	28	5	2354.0	2369.34	0.18	27
R_2_L_45	10	5	2338.0	2344.12	0.00	28	5	2345.4	2363.24	0.14	28
R_2_L_50	9	5	2527.2	2544.40	0.33	27	5	2542.1	2562.76	0.13	27
R_2_L_50	10	5	2525.4	2538.30	0.09	26	5	2536.8	2549.20	0.09	27
R_2_L_50	11	5	2525.4	2531.24	0.09	26	5	2533.4	2542.14	0.00	26

Table 9: Results for RC instances

Instance		ALNSM					ALNSP				
graph	m	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)
RC_1_S_35	8	1	2148.2	2148.20	2.77	32	0				
RC_1_S_35	9	5	1831.6	1849.74	0.04	30	5	1845.9	1879.88	0.51	30
RC_1_S_35	10	5	1786.8	1804.52	0.60	27	5	1800.7	1806.88	0.12	28
RC_1_S_40	9	5	2206.3	2234.56	1.56	29	5	2279.7	2317.84	0.41	29
RC_1_S_40	10	5	2177.3	2205.30	0.90	28	5	2212.9	2236.16	0.47	28
RC_1_S_40	11	5	2159.5	2179.32	1.88	27	5	2193.3	2203.56	1.84	26
RC_1_S_45	9	5	2241.5	2269.50	1.58	25	5	2273.3	2306.42	0.69	26
RC_1_S_45	10	5	2221.6	2245.28	0.66	25	5	2243.8	2275.24	1.39	25
RC_1_S_45	11	5	2205.8	2231.72	1.08	22	5	2234.8	2244.80	2.37	23
RC_1_S_50	11	5	2750.5	2774.88	1.34	28	5	2757.2	2811.46	0.85	28
RC_1_S_50	12	5	2699.7	2715.04	1.79	28	5	2685.1	2704.28	1.48	27
RC_1_S_50	13	5	2686.2	2697.92	1.81	23	5	2704.1	2724.02	2.28	24
RC_1_M_35	7	5	1672.1	1688.30	0.35	28	5	1674.3	1711.68	0.47	28
RC_1_M_35	8	5	1656.8	1664.68	0.00	28	5	1664.3	1683.58	0.00	30
RC_1_M_35	9	5	1640.8	1649.78	0.05	28	5	1658.4	1661.52	0.12	28
RC_1_M_40	7	4	1986.4	2031.33	0.00	28	4	2000.8	2023.13	1.11	26
RC_1_M_40	8	5	1937.4	1962.12	0.00	26	5	1971.8	1988.56	0.09	27
RC_1_M_40	9	5	1942.9	1949.28	0.25	27	5	1955.2	1970.22	0.00	28
RC_1_M_45	7	1	2070.4	2070.40	0.23	27	0				
RC_1_M_45	8	5	2057.9	2077.50	0.00	26	5	2073.1	2092.06	0.06	25
RC_1_M_45	9	5	2033.3	2050.36	0.04	25	5	2048.1	2055.70	0.04	24
RC_1_M_50	8	2	2495.8	2524.50	0.28	28	1	2614.1	2614.10	0.64	28
RC_1_M_50	9	5	2484.5	2497.08	0.18	27	5	2488.7	2499.06	0.22	27
RC_1_M_50	10	5	2451.0	2464.52	0.53	25	5	2478.5	2486.26	0.53	27
RC_1_L_35	6	5	1530.4	1539.52	0.00	28	5	1535.0	1546.12	0.00	28
RC_1_L_35	7	5	1536.4	1540.36	0.00	28	5	1538.7	1545.22	0.00	27
RC_1_L_35	8	5	1538.8	1542.66	0.00	27	5	1536.0	1543.56	0.00	27
RC_1_L_40	7	5	1791.4	1805.74	0.09	26	5	1790.6	1809.50	0.00	25
RC_1_L_40	8	5	1782.9	1798.56	0.00	25	5	1788.6	1811.86	0.00	25
RC_1_L_40	9	5	1794.8	1803.52	0.00	24	5	1808.7	1819.74	0.00	24
RC_1_L_45	7	5	1878.3	1886.44	0.15	25	5	1886.5	1900.24	0.15	25
RC_1_L_45	8	5	1885.8	1890.02	0.15	24	5	1881.6	1900.26	0.18	24
RC_1_L_45	9	5	1865.6	1884.86	0.15	23	5	1878.5	1883.82	0.15	24
RC_1_L_50	8	5	2240.5	2248.90	0.00	27	5	2245.8	2274.04	0.00	26
RC_1_L_50	9	5	2243.1	2254.28	0.00	26	5	2249.1	2266.48	0.00	25
RC_1_L_50	10	5	2235.5	2244.36	0.00	25	5	2256.2	2271.26	0.00	26
RC_1_S_35	9	5	2405.9	2424.34	0.77	31	5	2419.7	2455.16	0.72	31
RC_1_S_35	10	5	2397.2	2407.64	1.48	30	5	2407.4	2422.06	0.77	31
RC_1_S_35	11	5	2388.6	2397.44	0.93	28	5	2411.5	2426.42	0.64	30
RC_2_S_40	9	5	2448.1	2465.64	1.14	32	5	2480.5	2509.56	2.30	34
RC_2_S_40	10	5	2419.7	2427.46	0.78	32	5	2423.9	2444.48	1.30	34
RC_2_S_40	11	5	2384.0	2394.22	0.78	33	5	2407.5	2420.00	0.71	31
RC_2_S_45	10	5	2747.5	2760.12	2.49	31	5	2779.6	2791.88	2.33	33
RC_2_S_45	11	5	2662.6	2674.74	1.07	30	5	2678.3	2700.92	0.24	29
RC_2_S_45	12	5	2636.3	2646.78	1.67	28	5	2663.7	2677.90	2.55	27
RC_2_S_50	10	5	2976.8	2986.32	2.18	30	5	2993.8	3022.44	0.71	30
RC_2_S_50	11	5	2936.0	2949.46	0.62	29	5	2955.6	2984.92	1.38	29
RC_2_S_50	12	5	2905.7	2927.40	0.72	27	5	2906.1	2943.76	1.36	28
RC_2_M_35	8	5	2095.6	2107.00	0.33	29	5	2105.0	2127.78	0.55	29
RC_2_M_35	9	5	2088.8	2095.82	0.73	29	5	2087.0	2108.24	0.33	28
RC_2_M_35	10	5	2073.1	2086.36	0.42	28	5	2107.3	2116.78	0.33	28
RC_2_M_40	8	5	2144.8	2150.18	0.07	30	5	2152.1	2170.04	0.42	31
RC_2_M_40	9	5	2133.7	2137.26	0.53	29	5	2143.0	2157.34	0.75	31
RC_2_M_40	10	5	2120.0	2127.70	0.08	28	5	2124.3	2135.76	0.44	29
RC_2_M_45	8	4	2394.1	2407.88	0.00	29	3	2469.5	2518.33	1.24	29
RC_2_M_45	9	5	2361.9	2366.62	0.11	29	5	2366.3	2386.88	0.00	29
RC_2_M_45	10	5	2337.8	2355.12	0.33	30	5	2367.3	2379.58	0.15	28
RC_2_M_50	9	5	2560.5	2576.34	1.09	28	5	2586.0	2602.28	0.75	28
RC_2_M_50	10	5	2565.8	2580.54	0.85	28	5	2578.8	2601.62	2.76	27
RC_2_M_50	11	5	2561.1	2577.52	0.86	26	5	2576.2	2590.00	0.87	26
RC_2_L_35	7	5	1873.0	1882.10	0.02	29	5	1900.6	1908.52	0.07	30
RC_2_L_35	8	5	1878.1	1884.38	0.02	28	5	1869.7	1885.00	0.06	29
RC_2_L_35	9	5	1871.9	1876.60	0.02	28	5	1881.1	1892.34	0.02	27
RC_2_L_40	7	5	1882.8	1891.36	0.63	29	5	1903.4	1924.54	0.00	30
RC_2_L_40	8	5	1881.9	1886.68	0.42	29	5	1884.9	1888.68	0.61	29
RC_2_L_40	9	5	1881.1	1885.14	0.61	28	5	1875.1	1896.04	0.56	29
RC_2_L_45	7	2	2119.7	2131.55	1.51	28	0				
RC_2_L_45	8	5	2065.2	2084.00	0.00	28	5	2078.1	2095.82	0.00	28
RC_2_L_45	9	5	2076.1	2084.00	0.06	28	5	2064.4	2088.58	0.00	28
RC_2_L_50	8	5	2235.8	2244.70	0.19	26	5	2239.8	2282.88	0.19	27
RC_2_L_50	9	5	2217.1	2224.12	0.19	27	5	2215.0	2234.52	0.19	27
RC_2_L_50	10	5	2221.8	2225.98	0.54	26	5	2232.0	2236.14	0.00	27

Table 10: Results for C instances

Instance		ALNSM					ALNSP				
graph	m	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)	feas	Z^*	$\bar{Z}$	%wait (best)	trips (best)
C_1_S_45	12	4	2883.2	2901.60	4.36	27	4	2897.6	3015.80	1.26	26
C_1_S_45	13	5	2787.6	2797.06	2.89	26	5	2805.7	2820.54	2.07	27
C_1_S_45	14	5	2774.6	2779.04	2.58	26	5	2788.1	2803.76	1.85	25
C_1_M_35	9	5	2231.2	2253.68	0.57	28	5	2210.0	2269.24	2.30	30
C_1_M_35	10	5	2139.3	2177.80	0.01	27	5	2176.0	2205.56	0.68	26
C_1_M_35	11	5	2126.3	2135.02	0.73	25	5	2155.6	2171.64	0.02	27
C_1_M_45	8	2	2543.5	2561.20	0.99	30	0				
C_1_M_45	9	5	2435.5	2449.54	2.00	28	5	2437.1	2467.86	2.78	29
C_1_M_45	10	5	2394.3	2409.34	1.18	27	5	2410.5	2439.58	2.39	26
C_1_M_50	9	3	2882.7	2940.63	1.93	30	0				
C_1_M_50	10	5	2626.4	2670.38	0.89	28	5	2657.3	2857.46	1.29	26
C_1_M_50	11	5	2561.1	2585.44	0.31	28	5	2587.3	2611.44	0.64	27
C_1_L_35	7	5	1944.7	1951.50	0.02	27	5	1954.0	1965.52	0.00	28
C_1_L_35	8	5	1940.2	1949.82	0.00	27	5	1950.1	1958.18	0.00	27
C_1_L_35	9	5	1945.0	1950.28	0.00	27	5	1953.8	1962.66	0.01	27
C_1_L_40	6	1	1600.0	1600.00	0.00	22	0				
C_1_L_40	7	5	1807.3	1812.18	0.00	26	5	1800.7	1819.94	0.46	26
C_1_L_40	8	5	1789.5	1795.58	0.00	25	5	1801.6	1810.02	0.00	26
C_1_L_45	8	5	2138.1	2143.50	0.00	27	5	2140.5	2150.40	0.00	27
C_1_L_45	9	5	2130.1	2137.68	0.05	26	5	2146.9	2154.00	0.00	27
C_1_L_45	10	5	2134.5	2140.52	0.05	26	5	2134.4	2145.96	0.00	25
C_1_L_50	8	5	2334.0	2341.50	0.05	27	5	2335.7	2363.76	0.31	27
C_1_L_50	9	5	2323.3	2328.44	0.13	27	5	2331.1	2357.60	0.20	27
C_1_L_50	10	5	2313.5	2324.34	0.13	26	5	2331.4	2348.24	0.00	27
C_2_S_35	10	4	2374.7	2415.80	3.37	25	4	2432.9	2455.68	6.13	28
C_2_S_35	11	5	2305.8	2323.78	3.53	26	5	2320.9	2337.16	3.07	25
C_2_S_35	12	5	2293.8	2307.64	5.17	25	5	2304.7	2323.78	2.70	25
C_2_S_40	11	4	2944.0	2958.33	2.40	34	1	3075.6	3075.60	1.43	37
C_2_S_40	12	5	2806.9	2824.52	2.38	32	5	2836.5	2851.52	1.87	34
C_2_S_40	13	5	2777.8	2786.84	1.15	31	5	2817.0	2834.12	1.21	32
C_2_S_50	13	2	3319.5	3332.60	2.83	30	2	3378.6	3428.95	3.34	33
C_2_S_50	14	5	3157.6	3193.52	1.58	28	5	3208.0	3418.60	2.01	29
C_2_S_50	15	5	3155.3	3174.90	2.81	27	5	3170.1	3256.46	5.97	27
C_2_M_35	7	3	2007.1	2019.20	0.09	27	2	2022.3	2049.30	0.64	26
C_2_M_35	8	5	1885.6	1910.90	1.51	24	5	1893.3	1912.22	1.12	24
C_2_M_35	9	5	1878.2	1883.62	0.36	24	5	1889.9	1907.94	1.20	24
C_2_M_40	9	5	2525.4	2535.84	0.00	33	5	2520.9	2568.82	0.00	32
C_2_M_40	10	5	2473.2	2479.74	0.63	31	5	2483.9	2516.16	0.00	32
C_2_M_40	11	5	2445.1	2448.54	0.00	28	5	2465.3	2485.86	0.50	30
C_2_M_45	10	3	3300.3	3309.80	0.47	32	0				
C_2_M_45	11	5	3098.3	3119.58	0.48	33	5	3106.5	3144.02	0.34	33
C_2_M_45	12	5	3080.5	3086.12	1.16	30	5	3111.6	3125.88	0.43	32
C_2_M_50	9	5	2771.1	2825.22	0.36	29	3	2741.4	2864.33	2.16	28
C_2_M_50	10	5	2655.0	2676.76	0.38	26	5	2669.1	2698.42	0.91	27
C_2_M_50	11	5	2613.8	2621.48	0.05	26	5	2614.5	2639.34	0.15	25
C_2_L_35	7	5	1666.5	1672.36	0.00	24	5	1664.7	1675.96	0.00	24
C_2_L_35	8	5	1647.1	1661.08	0.00	24	5	1666.2	1671.16	0.00	23
C_2_L_35	9	5	1645.7	1663.60	0.09	22	5	1662.8	1669.30	0.00	22
C_2_L_40	8	5	2181.6	2184.60	0.59	30	5	2183.3	2194.32	0.68	30
C_2_L_40	9	5	2153.4	2162.32	0.72	30	5	2167.0	2177.48	0.00	30
C_2_L_40	10	5	2143.0	2150.14	0.69	29	5	2152.8	2165.74	0.00	29
C_2_L_45	9	5	2749.6	2771.38	0.36	30	5	2788.9	2805.40	0.00	30
C_2_L_45	10	5	2738.2	2745.18	0.04	30	5	2752.4	2766.46	0.01	30
C_2_L_45	11	5	2718.3	2729.88	0.00	28	5	2736.1	2748.80	0.10	29
C_2_L_50	8	5	2278.8	2290.92	1.25	27	5	2314.9	2345.28	0.44	27
C_2_L_50	9	5	2264.6	2270.92	0.97	26	5	2257.1	2283.74	0.35	26
C_2_L_50	10	5	2252.7	2264.24	0.29	26	5	2259.0	2265.44	0.29	25

Table 11 provides a summary of the average performance of the ALNSP algorithm compared to one of the ALNSM algorithm per type of instance. For each instance, we first compute the deviation $(\bar{Z}_{ALNSP_D} - \bar{Z}_{ALNSM_D})/\bar{Z}_{ALNSM_D}$. We then aggregate these results by computing the average of these deviations over each instance class and report the results in percentages. The ALNSP algorithm is clearly outperformed by the ALNSM algorithm on all classes of instances. Smaller gaps are observed for instances with large time windows, where assigning trips to vehicles is easier, due to the fact that trips may often be shifted in time while conserving their minimum duration.

The obtained solutions consistently contain multiple trips: when considering the best solutions produced respectively by the ALNSM and ALNSP for each instance, the average number of trips per vehicle over all instances is equal to 3.0 in both cases. This is significantly higher than the values obtained for the existing benchmarks. The average number of customers per trip is 3.7 in both cases, and the average number of customers per vehicle over all instances is 11.0.

For each instance, the total available working time is equal to 480 times the number of vehicles. On average over all instances, the ratio of the working duration in the best solution to the total available working time is equal to 0.79 for both algorithms. For the set of instances where the number of vehicles is the smallest for which we could find a feasible solution, this value is equal to 0.88 since the vehicle usage increases due to fleet size limitation. All other instance parameters being equal, the ratio also increases when the size of the time windows decreases. This makes sense since tight time windows provoke a significant waiting time increase as emphasized in the next section.

Table 11: Percentage gap of $\bar{Z}_{ALNSP_D}$ compared to $\bar{Z}_{ALNSM_D}$, aggregated per instance type

% Gap on average	Short planning horizon			Long planning horizon		
	S	M	L	S	M	L
R	0.87	1.03	0.44	0.87	1.15	0.89
RC	1.22	0.97	0.56	1.08	1.17	0.74
C	1.89	1.96	0.69	2.45	1.10	0.73

3.4 Total Duration Versus Total Travel Time

For each instance I, the best solution found over five runs with the $ALNSM_T$ configuration that minimizes the total travel time is compared to the best solution found over five runs with the $ALNSM_D$ configuration that minimizes the total duration. Let $\mathcal{S}_D(I)$ and $\mathcal{S}_T(I)$ be the best solutions yielded by $ALNSM_D$ and $ALNSM_T$ respectively, for a given instance I. Let $z(I)$ denote the variable part of the objective function value of $\mathcal{S}_D(I)$, i.e., its total travel time and waiting time. In this section, we report all indicators as percentages of $z(I)$ to assess how the decrease in travel time obtained with configuration $ALNSM_T$ impacts the waiting time independently of the order of magnitude of objective function values. For each

instance I, we calculate

- the difference ΔTT between the total travel time of $\mathcal{S}_T(I)$ and $\mathcal{S}_D(I)$,
- the difference ΔWT between the total waiting time of $\mathcal{S}_T(I)$ and $\mathcal{S}_D(I)$,
- the ratio $\Delta WT/\Delta TT$.

In Table 12, we provide a numerical example for instance $I = R\_1\_S\_50$. $\mathcal{S}_D(I)$ and $\mathcal{S}_T(I)$ are described in terms of their travel time and waiting time. In this case, the change in travel time ΔTT is equal to $-203.4 * 100/2768.9 = -7.35$ percent of $z(I)$ and the resulting change in waiting time ΔWT is equal to $819 * 100/2768.9 = 29.58$ percent of $z(I)$. The ratio $\Delta WT/\Delta TT$ is $29.58/(-7.35) = -4.02$, which indicates that, in this case, one unit of travel time decrease induces four units of waiting time increase.

Table 12: Example of instance $I = R\_1\_S\_50$, $z(I) = 2768.9$

	$\mathcal{S}_D(I)$			$\mathcal{S}_T(I)$			Difference	
	TT_D	WT_D	$TT_D + WT_D$	TT_T	WT_T	$TT_T + WT_T$	ΔTT	ΔWT
Absolute values	2731.1	37.8	2768.9	2527.7	856.8	3384.5	-203.4	819
Percentages of $z(I)$	98.63	1.37	100	91.29	30.94	122.23	-7.35	29.58

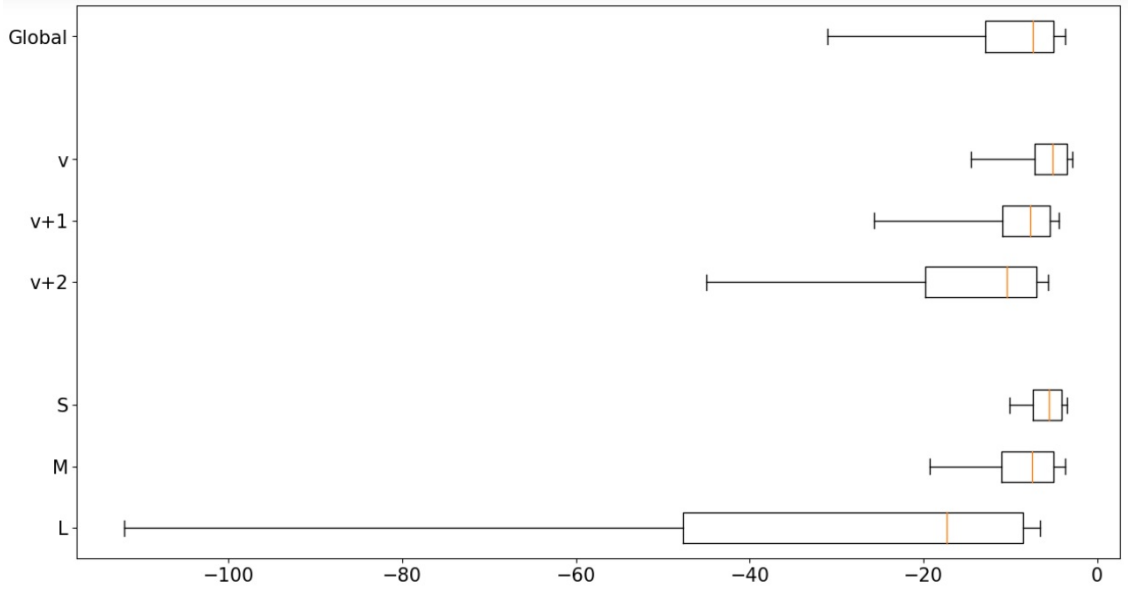
For a given instance I, ΔTT and ΔWT indicate the order of magnitude (with respect to $z(I)$) of the travel time decrease and of the waiting time increase in $\mathcal{S}_T(I)$ compared to $\mathcal{S}_D(I)$. The ratio $\Delta WT/\Delta TT$ is an indication of how the decrease in travel time of $\mathcal{S}_T(I)$ compared to $\mathcal{S}_D(I)$ impacts the waiting time. The more negative the ratio, the more important the waiting time increase induced by one unit of travel time decrease. We aggregate the values of ΔWT, ΔTT, and $\Delta WT/\Delta TT$ by computing their respective medians over all except 30 instances whose $\Delta WT/\Delta TT$ ratio is either not defined or positive. This ratio is not defined when at least one of the configurations could not find any feasible solution. It is positive when the $ALNSM_T$ configuration could not improve the travel time of the $ALNSM_D$ best solution. We use the median, a robust indicator, since the distribution of the ratio $\Delta WT/\Delta TT$ is skewed.

In Figure 2, boxplots yield the distribution of ratios $\Delta WT/\Delta TT$ for all the considered instances and for different instance categories. The medians are in orange, the boxes show the second and third quartiles, and the whiskers spread from the 10^{th} to the 90^{th} percentile. Table 13 shows the values of the medians $\mathcal{M}(\Delta WT)$, $\mathcal{M}(\Delta TT)$, and $\mathcal{M}(\Delta WT/\Delta TT)$ aggregated for all considered instances and per instance category.

Table 13: $\mathcal{M}(\Delta WT)$, $\mathcal{M}(\Delta TT)$ and $\mathcal{M}(\Delta WT/\Delta TT)$ per instance category

		Fleet size (m)			Time window size		
	Global	v	$v+1$	$v+2$	S	M	L
$\mathcal{M}(\Delta WT)$	32.86	15.61	31.00	49.27	49.03	29.80	23.85
$\mathcal{M}(\Delta TT)$	-4.16	-3.54	-4.49	-5.43	-8.05	-3.55	-0.89
$\mathcal{M}(\Delta WT/\Delta TT)$	-7.44	-5.12	-7.74	-10.34	-5.53	-7.44	-17.32

Figure 2: Percentiles 10-25-50-75-90 for the ratio $\mathcal{M}(\Delta WT/\Delta TT)$ per instance category



Global Observations. Let us first concentrate on the global indicators. $\mathcal{M}(\Delta WT/\Delta TT)$ is equal to -7.44: at the median, a travel time decrease of 1% of $z(I)$ leads to a waiting time increase of more than 7% of $z(I)$. Clearly, a small decrease in travel time is accompanied with a large increase in the waiting time. The median waiting time increase is 32.86% compared to a median travel time decrease of 4.16% which clearly shows that the additional waiting time of $\mathcal{S}_T(I)$ compared to $\mathcal{S}_D(I)$ is very important and impacts notably the total duration of the solution. Even if the waiting time increase in $\mathcal{S}_T(I)$ is expected, what is interesting is the magnitude of this effect.

Impact of the Fleet Size. As it can be seen from Figure 2 and Table 13, the fleet size impacts the effect described in the former paragraph. For the group of instances with an extremely tight number of vehicles, ΔWT is already quite high with a median value of 15.61 percent of $z(I)$, showing that sizing the fleet properly is not always sufficient to ensure a small waiting time. When the fleet size increases, the total working duration increases as well, creating flexibility and an opportunity to reduce the travel time at the cost of a further waiting time increase. This explains why $\mathcal{M}(\Delta WT)$ increases while $\mathcal{M}(\Delta TT)$ decreases with the fleet size. Note that the median waiting time increase of $\mathcal{S}_T(I)$ is very large when $m = v + 2$: 49.27% of $z(I)$. Also, $\mathcal{M}(\Delta WT/\Delta TT)$ decreases with the fleet size, indicating that the waiting time increase induced by one unit of travel time decrease becomes larger when the number of vehicles is augmented. Indeed, even if more flexibility is created by increasing the fleet size, obtaining a further travel time decrease becomes more and more expensive in terms of waiting time.

Impact of the Time Window Size. When time windows are large, the waiting time in $\mathcal{S}_D(I)$ is relatively small and configuration $\mathcal{S}_T(I)$ generally yields a very small travel time decrease (0.89%). In

counterpart, the waiting time increases very significantly, leading to large negative values of $\Delta WT/\Delta TT$. Consequently, the ratios $\Delta WT/\Delta TT$ spread over a larger range as the time window size increases. The median waiting time increase $\mathcal{M}(\Delta WT)$ is very large for instances with small time windows: 49.03% of $z(I)$. For such instances, when minimizing the total duration, concessions have to be made in terms of travel time to maintain an acceptable level of waiting time. This arbitrage becomes less and less necessary as the time windows are getting larger. Consequently, $\mathcal{M}(\Delta TT)$ increases while $\mathcal{M}(\Delta WT)$ decreases as the time window size increases. This causes a decrease of $\mathcal{M}(\Delta WT/\Delta TT)$: it becomes more and more difficult to obtain further travel time reductions when time windows are getting larger.

4 Conclusions

In this work, we tested two distinct methods to explore the search space of the MTVRPTW, which is a very relevant problem in practice, especially in the field of city logistics. The first method is an integrated approach using multi-trip operators that work directly on relaxed MTVRPTW solutions. The second one is a routing-packing decomposition approach that works on relaxed VRPTW solutions and subsequently assigns the created trips to available vehicles. Both methods integrate the most recent developments to explore efficiently the search space while relaxing time-related constraints.

The algorithms were configured using `irace`, an automatic configuration tool. ALNS algorithms are naturally parameter demanding. Moreover, we considered various algorithmic design choices. Hence, we employed `irace` to determine suitable values for the defined parameters instead of fixing them a priori. This methodology allowed us to compare the best possible implementation of our methods given the algorithmic components that we developed.

We first validated our approaches by comparing their results with those obtained on related problems by previous authors and concluded that they are both very effective. Then, instead of modifying Solomon's instances to fit the characteristics of our problem, we proposed new instances more adapted to the MTVRPTW context. We also provided a dedicated instance generator. We showed that the integrated multi-trip approach outperforms the decomposition approach in the presence of time windows and that the gap between both methods increases as the size of the time windows decreases. This makes sense since the assignment of a given set of trips to the available vehicles is more difficult to achieve in constrained problems. In contrast, the integrated approach is able to explicitly account for the fact that each route is composed of multiple trips, and to better take advantage of this feature. Knowing that an integrated approach is a totally valid alternative to routing-packing decomposition approaches is important for real-life applications where the decomposition approach may struggle due to side constraints.

We performed experiments with two different objective functions, the travel time and the duration, using the integrated approach. The aim here is not to create a new problem variant but rather to propose a reflection about the relevance of minimizing the total travel time in the presence of time windows. We

showed how, when minimizing the total duration, accepting a small increase in the travel time generally results in a significant decrease of the waiting time. This is an important observation since in practice, waiting times generate costs and drawbacks in transportation systems. We conclude that the total duration is a very relevant objective that deserves more attention in the scientific literature.

We believe that the methodological aspects emphasized in this work can benefit the vehicle routing research community if more broadly applied in future studies. In particular, it is a well-known fact that the set of training instances should always be distinct from the set of benchmark instances. This allows to obtain configurations that are not only good at solving benchmark instances but also at solving yet unseen instances, which makes also sense in real applications. However, it is a very common practice to parameterize vehicle routing algorithms on a subset of the benchmark instances, leading to biased results. Also, we believe that, in order to contribute to the state of knowledge, questioning and analyzing the commonly used model characteristics and the benchmark instances related to a given problem is at least as important as providing small improvements in the best know results. Finally, we insist on the importance of automatic algorithm configuration not only to fix the numerical parameter values but also to explore extensively the possible design options.

Acknowledgments

Computational resources were provided by the Consortium des Equipements de Calcul Intensif (CECI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11. The project leading to these results was partially supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, Grant P7/36. We thank the editors and the reviewers for their helpful comments.

Appendix

In this appendix, the creation process of new instances is detailed. Instance names are composed of four parts:

- a letter describing the location layout R, RC, or C,
- a number denoting the size of the planning horizon (1 for short and 2 for large),
- a letter denoting the time windows size S, M, or L,
- a number equal to the the radius of the instance (see below).

Depot. The x and y coordinates of the depot are both generated as random numbers between 0 and 10. The planning horizon is determined by the time window of the depot. Short planning horizons

are defined by a time window of $[0, 600]$ while long planning horizons are defined by a time window of $[0, 1200]$.

Vehicles. Vehicles have a capacity of 100 and a maximum shift duration D^{max} of 480.

Customer Locations. Customers are located according to three strategies:

- **Random (R) layout**: the repartition of the customers is uniform inside a circle with radius $r \in 35, 40, 45, 50$ which is always the last part of the instance name. A squared distance is generated in $\mathcal{U}(0, r^2)$ and an angle in $\mathcal{U}(0, 2\pi)$. Euclidean coordinates are deduced and rounded to the first decimal place.
- **Clustered (C) layout**: 10 to 12 cluster centers are generated exactly with the same process as the one described above. If one of these centers is generated close to another one (distance smaller than $r/2.5$), a new attempt is made to generate it elsewhere. When all cluster centers are located, customers are generated one by one as described below.
  1. Assign randomly the customer to one of the clusters.
  2. Generate sd in $\mathcal{U}(2, 4)$.
  3. Generate an angle in $\mathcal{U}(0, \pi)$ and a distance in a truncated normal distribution $\mathcal{N}(-sd, sd)$ defined between $-0.5sd$ and $0.5sd$. The generated angle and distance represent the coordinates of the customer relatively to the cluster center.
  4. Compute the Euclidean coordinates of the customer and round them to the first decimal place.
- **Random and clustered (RC) layout**: 60% of the customers coordinates are computed as in the R layout and 40% as in the C layout. Note that some random customers will likely be generated inside clusters. For the clustered customers, 5 or 6 cluster centers are generated instead of 10 to 12 and sd is generated in $\mathcal{U}(3, 4)$ instead of $\mathcal{U}(2, 4)$.

When Euclidean distances are computed to obtain travel times from the Euclidean coordinates in the instance files, they are rounded to the first decimal place.

Customer Demands. Customer demands are generated randomly with a beta distribution $\beta(1.6, 5)$. The obtained numbers are in $[0, 1]$ and so we project them on $[1, ..., 100]$.

Service Times. The service time is equal to 10 for all customers.

Customer Time Windows. For each VRP graph generated, three time window sizes are considered: 60 (S), 120 (M), 240 (L). For a given graph, we generate the time window center of customer i as $u + t_{0i}$ where $u \sim \mathcal{U}(2, b_0 - a_0 - 2t_{0i} - s_i)$. Then, each time window is created around its center depending on the desired size ($-30/+30$ for S, $-60/+60$ for M, $-120/+120$ for L). Note that reaching the time

window early or leaving it late might not be feasible in some cases but the center of each time window is always reachable.

References

Azi, N., Gendreau, M., and Potvin, J.-Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, 41:167–173.

Battarra, M., Monaci, M., and Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, 36:3041–3050.

Cattaruzza, D., Absi, N., and Feillet, D. (2016a). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50:676–693.

Cattaruzza, D., Absi, N., and Feillet, D. (2016b). Vehicle routing problems with multiple trips. *4OR - A Quarterly Journal of Operations Research*, 14:223–259.

Cattaruzza, D., Absi, N., Feillet, D., and González-Feliu, J. (2017). Vehicle routing problems for city logistics. 6(1):51–79.

Cattaruzza, D., Absi, N., Feillet, D., and Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236:833–848.

Comité National Routier (2017). Indices et statistiques.

Despaux, F. and Basterrech, S. (2016). Multi-trip vehicle routing problem with time windows and heterogeneous fleet. *International Journal of Computer Information Systems and Industrial Management Applications.*, 8:355 – 363.

Fleischmann, B. (1990). The vehicle routing problem with multiple use of vehicles. Technical report, Fachbereich Wirtschaftswissenschaften, Universität Hamburg.

François, V., Arda, Y., Crama, Y., and Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, 255(2):422–441.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2013). An exact algorithm to solve the multi-trip vehicle routing problem with time windows. Technical report.

Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2014). A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4OR*, 12:235–259.

Hernandez, F., Feillet, D., Giroudeau, R., and Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 249:551–559.

Kergosien, Y., Gendreau, M., and Billaut, J.-C. (2017). A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research*, 262(1):287 – 298.

Lee, J., Kim, B.-I., Johnson, A. L., and Lee, K. (2014). The nuclear medicine production and delivery problem. *European Journal of Operational Research*, 236(2):461 – 472.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., , Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43 – 58.

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

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

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435.

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

Ropke, S. and Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472.

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171:750–775.

Savelsbergh, M. (1992). The vehicle routing problem with time windows : minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98, Fourth international conference on principles and practice of constraint programming*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431.

Taillard, E., Laporte, G., and Gendreau, M. (1996). Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47:1065–1070.

Union Professionnelle du Transport et de la Logistique (2012). Cahier de revendications.

Vidal, T., Crainic, T., Gendreau, M., and Prins, C. (2015). Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics*, 21(3):329–358.

Wang, Z., Liang, W., and Hu, X. (2014). A metaheuristic based on a pool of routes for the vehicle routing problem with multiple trips and time windows. *Journal of the Operational Research Society*, 65:37–48.