Large neighborhood search for multi-trip vehicle routing

Véronique François[a], Yasemin Arda[a], Yves Crama[a], Gilbert Laporte[b]

[a] *QuantOM, HEC Management School, University of Liège, Belgium*
[b] *Canada Research Chair in Distribution Management, HEC Montréal, Canada*

Abstract

We consider the multi-trip vehicle routing problem, in which each vehicle can perform several routes during the same working shift to serve a set of customers. The problem arises when customers are close to each other or when their demands are large. A common approach consists of solving this problem by combining vehicle routing heuristics with bin packing routines in order to assign routes to vehicles. We compare this approach with a heuristic that makes use of specific operators designed to tackle the routing and the assignment aspects of the problem simultaneously. Two large neighborhood search heuristics are proposed to perform the comparison. We provide insights into the configuration of the proposed algorithms by analyzing the behavior of several of their components. In particular, we question the impact of the roulette wheel mechanism. We also observe that guiding the search with an objective function designed for the multi-trip case is crucial even when exploring the solution space of the vehicle routing problem. We provide several best known solutions for benchmark instances.

Keywords: Transportation, Vehicle routing, Multi-trip, Large neighborhood search, Automatic algorithm configuration

1. Introduction

The multi-trip vehicle routing problem (MTVRP) is a variant of the vehicle routing problem (VRP) in which each vehicle can perform one or more trips, also called *routes*, during the same planning period. The set of routes performed by a given vehicle constitutes a *tour* whose total duration cannot exceed a given time limit. The MTVRP arises when the demands of the customers are large compared to vehicle capacities or when the distances are relatively short. In food distribution systems, drivers sometimes perform two, three or even more delivery routes during the same working day.

It is common to solve the MTVRP by combining VRP and bin packing (BP) algorithms (Fleischmann, 1990; Taillard et al., 1996; Petch and Salhi, 2004; Salhi and Petch, 2007; Olivera and Viera, 2007). Vehicle routes are first obtained by applying VRP algorithms. These routes are then assigned to a fleet with a limited number of vehicles, generally by applying BP techniques: each route is viewed as an item whose size corresponds to its duration and each vehicle as a bin of capacity equal to the maximum allowed tour duration.

The main contribution of this work is to propose local search operators specifically designed for multi-trip variants of the VRP. They stem from classical VRP operators and take into consideration the routing and route assignment aspects of multi-trip problems. Our aim is to compare the performance of algorithms that incorporate these multi-trip operators with those that treat the routing and the packing subproblems separately. Two adaptive large neighborhood search (ALNS) algorithms (Ropke and Pisinger, 2006a,b; Pisinger and Ropke, 2007) are described: the ALNS with multi-trip operators (ALNSM) and the ALNS combined with BP (ALNSP). Both algorithms are tested on the benchmark instances of Taillard et al. (1996). The ALNSM yields very good results but is outperformed by the ALNSP which produces 10 new best known MTVRP solutions.

As a second contribution, we analyze the behavior of various ALNS components and we describe the interactions between some of these. Every implementation option that was considered during the design phase of the algorithms is given, not only the most efficient one, thus providing insights into the global behavior of the proposed ALNS metaheuristics. We use the `irace` package (López-Ibáñez et al., 2011), an automatic configuration tool, not only as a fine-tuning engine, but also as a means to gain meaningful algorithmic insights.

Section 3 describes the MTVRP along with some notations. Section 4 presents the specific multi-trip operators. Section 5 details the ALNSM and ALNSP implementations. The results are then presented in Section 6, along with further experiments about algorithm components in Section 6.4. Section 7 presents the conclusions.

Email addresses: `veronique.francois@ulg.ac.be` (Véronique François), `yasemin.arda@ulg.ac.be` (Yasemin Arda), `yves.crama@ulg.ac.be` (Yves Crama), `gilbert.laporte@cirrelt.ca` (Gilbert Laporte)

2. Literature review

Fleischmann (1990) was the first to address the MTVRP. He combined a modified savings heuristic with a BP heuristic. Taillard et al. (1996) later proposed a three-phase method. First, a large number of VRP routes are generated by constructing and modifying VRP solutions using a tabu search (TS) algorithm. Several VRP solutions are then built based on the routes constructed in this first step. Finally, a BP procedure is applied to each VRP solution in an attempt to generate feasible MTVRP solutions. In Brandão and Mercer (1998), a constructive procedure sequentially builds an MTVRP solution that may involve overtime. A TS procedure then refines this solution by applying classical VRP operators. Petch and Salhi (2004) proposed a multi-phase construction algorithm in which VRP solutions are iteratively created and refined. Each is tentatively transformed through a BP procedure until a feasible MTVRP solution is found or a stopping criterion is satisfied. Salhi and Petch (2007) applied a genetic algorithm in which chromosomes represent ordered circular sectors. A savings heuristic is applied to solve VRPs within the circular sectors, and a BP heuristic then creates MTVRP solutions out of the routes produced by the savings procedure. Olivera and Viera (2007) proposed an adaptive memory programming (AMP) procedure combining TS and a BP heuristic. A memory of routes is initialized by constructing VRP solutions by means of the sweep algorithm. The algorithm then iteratively creates and improves new VRP solutions from routes that are randomly chosen in the memory. These VRP solutions are improved by applying a TS procedure. A BP heuristic is used at each iteration of the TS heuristic in order to tentatively produce a feasible MTVRP solution. The new routes are added to the adaptive memory and sorted according to the quality of the MTVRP solution to which they belong. Cattaruzza et al. (2014b) provided state-of-the-art results for the MTVRP. These authors proposed a memetic algorithm in which each chromosome defines a customer sequence. A modified split procedure (Prins, 2004) partitions the customer sequences to tentatively obtain an MTVRP solution. The authors described a second version of their algorithm in which a local search procedure reassigns routes to vehicles while at the same time performing a VRP move. Recently, Mingozzi et al. (2013) developed an exact algorithm for the MTVRP which yielded an optimal solution for 42 out of the 106 benchmark instances of Taillard et al. (1996). Their model combines a partitioning formulation with valid inequalities. For an extensive review of MTVRP variants, the interested reader is referred to Cattaruzza et al. (2014a).

3. Problem description and notations

Let $G(V, E)$ be a complete undirected graph where $V = \{0, ..., n\}$ is the set of nodes and $E = \{(i, j)|i, j \in V, i < j\}$ is the set of edges. Each node $i = 1, ..., n$ represents a customer, while node $i = 0$ represents the depot. With each customer $i = 1, ..., n$ is associated a demand d_i that must be satisfied by exactly one delivery (i.e., split deliveries are not allowed). A fleet of m identical vehicles is based at the depot. The travel time on edge $(i, j) \in E$ is t_{ij}. Each vehicle $k = 1, ..., m$ has a limited capacity Q and a maximum allowed working duration T_{max}, and must perform a tour $\mathcal{T}_k$ made up of a set of routes starting and ending at the depot. The total demand of the customers served by any route of $\mathcal{T}_k$ must not exceed Q, and the time needed to perform $\mathcal{T}_k$ must not exceed T_{max}. The objective is to determine a set of tours minimizing the total travel time while satisfying the constraints.

The metaheuristics that we have developed work on a relaxed version of the MTVRP called the R-MTVRP, in which the tour duration constraints are not considered. In the following, we denote MTVRP solutions by $\mathcal{S}$, and R-MTVRP solutions by $\hat{\mathcal{S}}$. If at least one vehicle of an R-MTVRP solution $\hat{\mathcal{S}}$ travels for a duration that exceeds T_{max}, then $\hat{\mathcal{S}}$ contains some overtime. The overtime of vehicle k, denoted by O_k, is defined as $O_k = \max\{0, D_k - T_{max}\}$, where D_k is the total duration of tour $\mathcal{T}_k$. The total overtime of $\hat{\mathcal{S}}$ is defined as $O_{\hat{\mathcal{S}}} = \sum_{k=1,...,m} O_k$. The objective function of the R-MTVRP includes the total travel time, as well as the penalized overtime. The associated penalization factor will be introduced later on. R-MTVRP solutions that do not contain overtime are also feasible MTVRP solutions.

Solutions to the well-known capacitated vehicle routing problem (CVRP) are used to explore the solution space during the execution of the ALNSP heuristic. The CVRP is defined on the same graph $G(V, E)$ as the MTVRP. The fleet size is unlimited, while each vehicle having a capacity Q can perform only one route. The objective function to be minimized is the total travel time. Let $\mathcal{X}$ be the set of routes that constitute a feasible CVRP solution. In order to transform $\mathcal{X}$ into an R-MTVRP solution, each route of $\mathcal{X}$ has to be assigned to one of the m available vehicles. All assignments of the routes in $\mathcal{X}$ that satisfy the MTVRP tour duration constraints yield MTVRP solutions that are equivalent in terms of their objective function value since they have the same duration and contain no overtime. Finding such a feasible assignment is equivalent to solving a BP problem where each route is an item with a size corresponding to its duration, and each vehicle is a bin of capacity equal to the maximum allowed tour duration. Since the CVRP fleet size is unlimited, if $\mathcal{X}$ is an optimal CVRP solution, then each feasible assignment of its routes provides an optimal MTVRP solution.

Table 1 presents the recurring notations of this paper.

Problem parameters	
m	Number of vehicles.
n	Number of customers to be served.
Q	Vehicle capacity.
T_{max}	Maximum allowed duration for tours.
Problem related notations	
$\mathcal{X}$	CVRP solution.
$\mathcal{S}$	MTVRP solution.
$\hat{\mathcal{S}}$	R-MTVRP solution.
$\mathcal{T}_k$	Tour performed by vehicle k.
D_k (resp., $D_{\hat{\mathcal{S}}}$)	Total duration of the tour performed by vehicle k (resp., of solution $\hat{\mathcal{S}}$).
O_k (resp., $O_{\hat{\mathcal{S}}}$)	Total overtime of vehicle k (resp., of solution $\hat{\mathcal{S}}$).
α	Total overtime penalty factor in R-MTVRP solutions.
Algorithm related notations	
ALNSM	ALNS with multi-trip operators.
ALNSP	ALNS combined with bin packing.
q	Number of requests removed and reinserted during an ALNS iteration.
H_{rem} and H_{ins}	Set of removal heuristics and set of insertion heuristics.
Algorithmic parameters	
α_{min} and α_{max}	Minimum and maximum allowed value for α.
μ	Adaptation factor for α.
ξ	Number of iterations without resetting the value of α.
θ_0 and η	Initial temperature and cooling factor of the simulated annealing framework.
κ	Factor used to set the minimum temperature as a fraction of θ_0.
σ_1, σ_2 and σ_3	Rewards influencing the heuristic selection process of the roulette wheel.
ρ and Θ	Persistence factor and time segment of the roulette wheel.
υ and δ	Determinants of the number of customers removed and reinserted in a given iteration.
p	Randomization factor of removal heuristics.
Φ	Relatedness measure parameter of Shaw removal heuristic.
λ_H and α_H	Overtime penalization factor and size of the historical memory.
γ and Γ	Parameters of the tabu-edge mechanism for insertion heuristics.
λ_M and α_M	Overtime penalization factor and size of the adaptive memory of routes.
y_M	Randomization factor for route selection in the adaptive memory of routes.
ψ	"Pre-acceptance" factor of solutions in the ALNSP algorithm.
$ALNS_{iter}$	Number of iterations of the ALNSP inner loop.
p_{US}	Determinant of the neighborhood size of the US procedure.

Table 1: Recurring notations

4. Specific local search operators

The purpose of the proposed multi-trip operators is to manage the routing and the assignment aspects of the problem simultaneously, instead of treating them independently. In MTVRP or R-MTVRP solutions, any reordering of the routes of a given tour $\mathcal{T}_k$ leaves its total duration and overtime unchanged. However, the operators presented below are not designed to treat routes as separate entities. Instead, they treat any tour $\mathcal{T}_k$ as a giant tour made up of the routes of vehicle k, i.e., an ordered sequence of routes. More precisely, the representation of a tour starts with an origin depot, ends with a destination depot, and contains the customer sequence of each of its routes. Each of these customer sequences is separated by a depot that we call an "internal depot". In the following, routes in $\mathcal{T}_k$ are said to be consecutive if their respective customer sequences are only separated by one internal depot in the giant tour representation of $\mathcal{T}_k$.

4.1. Removal and insertion operators

The specific removal and insertion operators presented below adapt to the MTVRP context destroy-and-repair heuristics (similar to those used by Ropke and Pisinger (2006a) and Pisinger and Ropke (2007)).

4.1.1. Inserting a customer in a multi-trip context

Let $\hat{\mathcal{S}}$ be a partial solution of a given R-MTVRP. In $\hat{\mathcal{S}}$, a sequence of nodes (depots and customers) forming a tour $\mathcal{T}_k$ is known for each available vehicle k, but some of the customers are not assigned to any vehicle. Some of the vehicles may also be empty (the representation of the associated tour contains only the origin and the destination depots). The solution $\hat{\mathcal{S}}$ needs to be repaired by inserting in it the unrouted customers. Let v_i and v_{i+1} be two consecutive nodes of a tour $\mathcal{T}_k = (..., v_i, v_{i+1}, ...)$ in $\hat{\mathcal{S}}$. We consider four possible schemes to insert a customer v_j between v_i and v_{i+1}:

- *Scheme 1*: $\mathcal{T}_k \leftarrow (..., v_i, v_j, v_{i+1}, ...)$,
- *Scheme 2*: $\mathcal{T}_k \leftarrow (..., v_i, v_j, 0, v_{i+1}, ...)$,
- *Scheme 3*: $\mathcal{T}_k \leftarrow (..., v_i, 0, v_j, v_{i+1}, ...)$,
- *Scheme 4*: $\mathcal{T}_k \leftarrow (..., v_i, 0, v_j, 0, v_{i+1}, ...)$.

Scheme 1 is the classical VRP insertion scheme. In scheme 2 (resp., 3), v_j is also inserted between v_i and v_{i+1} along with a stop at the depot after (resp., before) visiting v_j. In scheme 4, a new route containing a single customer v_j is created between v_i and v_{i+1}, i.e., two depots are inserted at the same time as customer v_j.

4.1.2. Removing a customer in a multi-trip context

Let v_{i-1}, v_i and v_{i+1} be three consecutive nodes of a given tour $\mathcal{T}_k = (..., v_{i-1}, v_i, v_{i+1}, ...)$ in an R-MTVRP solution $\hat{\mathcal{S}}$. When removing v_i, $\mathcal{T}_k$ becomes $\mathcal{T}_k \leftarrow (..., v_{i-1}, v_{i+1}, ...)$. If v_{i-1} and v_{i+1} are both depots, i.e., $v_{i-1} = v_{i+1} = 0$, one of these is removed along with v_i.

4.1.3. Merging consecutive routes

If capacity constraints allow it, consecutive routes of $\mathcal{T}_k$ may be merged together by removing the internal depot that separates their respective customer sequences. Here we apply a merge operator during the course of the ALNSM algorithm. Our specific implementation of this operator is described at the end of Section 5.3.1 that details our removal heuristics.

4.2. Multi-trip improvement operators

Most classical improvement neighborhood structures for the VRP are usually classified into two categories: node-based neighborhoods and edge-based neighborhoods. In node-based neighborhood structures, moves are performed by repositioning or exchanging nodes in a given solution, while edge-based neighborhood structures work with edge deletions and insertions. The difference between these two types of neighborhoods is often a matter of convenience. In this work, in order to adapt these classical neighborhood structures for the multi-trip case, internal depots are treated as if they were customers in a giant tour representation of each vehicle. Consider an example involving a typical node-based operator. Suppose that there are two vehicle tours $\mathcal{T}_k = (..., v_{i-1}, v_i, v_{i+1}, ...)$ and $\mathcal{T}_l = (..., v_{j-1}, v_j, v_{j+1}, ...)$. Repositioning v_i after v_j in $\mathcal{T}_l$ with the relocate operator leads to $\mathcal{T}_k \leftarrow (..., v_{i-1}, v_{i+1}, ...)$ and $\mathcal{T}_l \leftarrow (..., v_{j-1}, v_j, v_i, v_{j+1}, ...)$. Such a move can be considered even if v_i is a depot (i.e., $v_i = 0$), as for any customer. Applying such multi-trip operators based on the giant tour representation of each vehicle may result in having two consecutive depots in a tour, i.e., an empty route. Removing one of them deletes this empty route and does not further alter the giant tour structure.

A drawback of adapting these well-known VRP operators is that the number of internal depots of a given tour can only decrease or remain constant since moves can result in deleting consecutive depots. However, the multi-trip improvement operators serve only during the post-otpimization phase of the ALNSM. Hence, we do not consider increasing the number of depots.

5. Adaptive large neighborhood search

In order to compare the multi-trip operators with classical solution strategies involving routing and packing techniques, we implement both solution approaches within a common framework, namely the ALNS scheme of Ropke and Pisinger (2006a). It relies on a set of removal heuristics and a set of insertion heuristics which iteratively destroy and repair solutions. The selection probability of each heuristic at a given iteration is influenced by its performance during past iterations. Removal heuristics select customers to be removed based on criteria detailed in Section 5.3.1. Insertion heuristics use greedy or regret mechanisms described in Section 5.3.2.

The algorithm components described throughout this section include several numerical parameters, whose values are determined during a configuration phase. For this purpose, an automatic configuration tool, `irace` (López-Ibáñez et al., 2011), is used for both algorithms. Some of the features described in this section are optional, i.e., the configuration tool determines if they should be included or not in the final implementation of the methods. The configuration phase is further described in Section 6.2.

5.1. *Adaptive large neighborhood search with multi-trip operators (ALNSM)*

The ALNSM algorithm explores the R-MTVRP solution space by means of multi-trip operators.

5.1.1. *Initial solution*

The construction of the initial solution is handled by a greedy heuristic, described in Section 5.3.2, using the four multi-trip insertion schemes detailed in Section 5.3.2.

5.1.2. *Objective function*

As stated in Section 3, the objective function of the MTVRP is the total travel time which has to be minimized. However, for most instances, obtaining a feasible solution is a challenging task. Hence, R-MTVRP solutions, whose values contain both travel time and penalized overtime, are visited during the course of the ALNSM. Visiting R-MTVRP solutions with high overtime can be interesting because they can guide the search towards new regions of the solution space with shorter total durations.

The cost of a solution $\hat{\mathcal{S}}$ is defined as $C_{\hat{\mathcal{S}}} = \sum_k D_k + \alpha \sum_k O_k$, where $\alpha \in [\alpha_{min}, \alpha_{max}]$ is an adaptive parameter akin to the objective function penalties described in Olivera and Viera (2007). At the beginning of the ALNSM, α is set to α_{min}. A parameter $\mu \in [1, 2]$ is introduced to control the variation of the value of α. Each time a solution $\hat{\mathcal{S}}$ is accepted:

- either $\hat{\mathcal{S}}$ contains overtime and we would like to reduce it, so the value of α is set to $\min\{\alpha\mu, \alpha_{max}\}$,
- or $\hat{\mathcal{S}}$ contains no overtime and the search should focus on reducing the travel time, even if this results in overtime, so the value of α is set to $\max\{\alpha/\mu, \alpha_{min}\}$.

The values of α_{min}, α_{max} and μ are determined during the ALNSM configuration phase, and so is ξ, the number of iterations after which the value of α is reset to α_{min}. Resetting the value of the overtime penalty α prevents it from remaining stuck at its maximum value α_{max} while trying to find a feasible solution. Note that whereas the acceptance criterion is based on the above cost function which includes an adaptive overtime penalty factor α, improvement is in contrast evaluated by checking whether the overtime of the new solution $\hat{\mathcal{S}}'$ is smaller than the overtime of the incumbent solution $\hat{\mathcal{S}}$. If both overtimes are equal, then the total durations are compared. The same reasoning holds to detect a new best solution.

5.1.3. *Acceptance criterion*

When a new solution is created by destroying and reconstructing the incumbent solution, a simulated annealing criterion is used to determine whether the new solution is accepted. Let $\hat{\mathcal{S}}$ be the incumbent R-MTVRP solution and let $\hat{\mathcal{S}}'$ be the candidate examined for acceptance with $C_{\hat{\mathcal{S}}'} > C_{\hat{\mathcal{S}}}$, then $\Pr(\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}}') = \exp[-(C_{\hat{\mathcal{S}}'} - C_{\hat{\mathcal{S}}})/\theta C_{\hat{\mathcal{S}}}]$, 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 parameters that need to be configured are the initial temperature θ_0, the cooling factor η, and κ, which determines the minimum temperature as a function of the initial temperature, $\theta_{min} = \kappa\theta_0$.

5.1.4. *Destroy-and-repair*

At every iteration of the ALNSM, the incumbent solution $\hat{\mathcal{S}}$ is modified by removing q nodes from the solution and reinserting them. Removals and insertions are respectively performed by a set H_{rem} of removal heuristics and a set H_{ins} of insertion heuristics based on the multi-trip removal and insertion operators described in Section 4.1. The removal and insertion heuristics are described in Section 5.3. At the first iteration, q is equal to 1. During the course of the ALNSM, the variation of q is based on its current value and on the acceptance of solution $\hat{\mathcal{S}}$ created during the previous iteration:

- if $\hat{\mathcal{S}}$ is accepted, then $q = 1$,
- if $\hat{\mathcal{S}}$ is rejected and $q < q_{max}$, then $q \leftarrow q + 1$,
- if $\hat{\mathcal{S}}$ is rejected and $q = q_{max}$, then $q \leftarrow q_{low}$.

The value of q can never exceed $q_{max} = \lfloor \upsilon n \rfloor$, where υ is a parameter smaller than 1. The value q_{low} is defined as $\lfloor q_{max}/\delta \rfloor$, where δ is a parameter greater than 1. The rationale for using $q \leftarrow q_{low}$ instead of $q \leftarrow 1$ is based on the observation made in Ropke and Pisinger (2006a) that improvements rarely results from the use of very small values of q when no new solution has been accepted for many iterations.

5.1.5. *Roulette wheel*

Our roulette wheel mechanism is mostly the same as that of Ropke and Pisinger (2006a), except that we include an additional normalization process to facilitate the configuration of the roulette wheel parameters with `irace`. At each iteration of the ALNSM, weights w_h, where $h \in H_{rem} \cup H_{ins}$, are used to randomly select a removal heuristic $h_{rem} \in H_{rem}$ and an insertion heuristic $h_{ins} \in H_{ins}$. The weights are kept constant during a certain number of iterations Θ called a time segment. During the first time segment, all removal (resp., insertion) heuristics have the same weight and $\sum_{h \in H_{rem}} w_h = 1$ (resp., $\sum_{h \in H_{ins}} w_h = 1$). In each time segment, these weights are updated as explained below.

Each heuristic $h \in H_{rem} \cup H_{ins}$ has a score Ω_h reinitialized at zero at the beginning of every time segment. Whenever a solution $\hat{\mathcal{S}}'$ is accepted, the scores Ω_h of both heuristics h_{rem} and h_{ins} are updated: $\Omega_h \leftarrow \Omega_h + \sigma_\phi$, where ϕ is equal to 1 if $\hat{\mathcal{S}}'$ is a new best solution, 2 if $\hat{\mathcal{S}}'$ is an improved solution without being a new best one, and 3 otherwise. The values σ_ϕ belong to the interval $[0, 1]$ and are normalized to satisfy $\sigma_1 + \sigma_2 + \sigma_3 = 1$. At the end of a time segment, Ω_h is set to Ω_h / u_h, where u_h is the number of times heuristic h was used during the current time segment. If $u_h = 0$ (which may happen especially if the value of $\bar{\Omega}_h$ was small during former time segments), then we let Ω_h take the same value as in the previous time segment.

The scores are then normalized for H_{rem} and H_{ins}, respectively: $\sum_{h \in H_{rem}} \bar{\Omega}_h = 1$ and $\sum_{h \in H_{ins}} \bar{\Omega}_h = 1$, where $\bar{\Omega}_h$ is the normalized score of h. At the end of a time segment, the weight of each heuristic is updated as $w_h \leftarrow (1-\rho) w_h + \rho \bar{\Omega}_h$ for $h \in H_{rem} \cup H_{ins}$, where ρ is a parameter, and $(1 - \rho)$ represents the persistence of information from the previous segments.

The values of Θ, σ_1, σ_2, $\sigma_3 = 1 - \sigma_1 - \sigma_2$ and ρ are determined during the configuration phase.

5.1.6. *US local search*

The US improvement procedure of GENIUS algorithm (Gendreau et al., 1992) is applied optionally to improve each route of a solution. The configuration phase determines whether the US procedure is applied whenever a new best solution is found, or whenever a solution is accepted to be the new incumbent, or never. Only those routes that were modified since the last call of the procedure need to be optimized. A parameter p_{US} determines how many neighbors are scanned at each move.

5.1.7. *Stopping criterion*

The parameters of the algorithm are tuned by solving an "anytime" parameter optimization problem. This means that the parameters obtained at the end of the configuration phase are supposed to provide reasonably good results independently of the running time of the algorithm or of the number of ALNSM iterations. In Section 6, we report results for several stopping criteria. However, the general idea is the following: the algorithm stops after a given number of iterations $Iter$ if a feasible solution is found; else, the algorithm continues until a feasible solution has been found or until a time limit proportional to the square of the instance size has been reached.

5.1.8. *Post-optimization*

Once the stopping criterion of the ALNSM is met, a post-optimization procedure is performed to improve the best known solution. This procedure is a variable neighborhood descent (VND) (Hansen and Mladenović, 2001), i.e., a descent heuristic that uses a list of neighborhoods to improve the solution at hand. These neighborhoods are based on adaptations of well-known VRP operators, as described in Section 4.2. For the purpose of this post-optimization phase, the overtime penalty α is fixed at a very high value to ensure that two solutions containing overtime are first compared on this basis. If no overtime remains, then travel times are compared. Indeed, the primary goal is to decrease the remaining overtime if any. However, if a feasible MTVRP solution is found, the post-optimization phase focuses on decreasing its total duration while preserving feasibility. The post-optimization VND explores the neighborhoods in the following order:

- the 2-opt neighborhood that treats each vehicle's giant tour independently,
- the relocate neighborhood that relocates a chain of nodes from its current position to another position in the same or in another vehicle,
- the exchange neighborhood that swaps two chains of nodes visited by the same or by different vehicles,
- the US procedure that re-optimizes each route separately.

The relocate and the exchange neighborhoods both treat a maximum chain length of four nodes. Neighborhoods with chains of increasing length are sequentially treated as different neighborhoods. At the end of the post-optimization phase, if there exist chains containing two consecutive depots, one of these is deleted.

5.1.9. Method summary

The ALNSM method is summarized below as it was implemented for the configuration phase. Some of the design options are determined by using `irace` as further explained in Section 6.2.

Algorithm 1 ALNSM

```
1: Construct an initial R-MTVRP solution $\hat{\mathcal{S}}$ by using an insertion heuristic based on the four multi-trip insertion
   schemes
2: $\hat{\mathcal{S}}_{best} \leftarrow \hat{\mathcal{S}}$
3: $q = 1$; initialize the roulette wheel; initialize the adaptive parameters
4: while Stopping criterion is not met do
5:     Roulette wheel: select a removal heuristic $h_{rem}$ and an insertion heuristic $h_{ins}$
6:     Remove customers from $\hat{\mathcal{S}}$ using $h_{rem}$, creating a partial solution
7:     Insert customers into the partial solution using $h_{ins}$, creating a solution $\hat{\mathcal{S}}'$
8:     if Acceptance criterion is met then
9:         Optional: apply the US procedure on each route of $\hat{\mathcal{S}}'$
10:        $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}}'$
11:        $q \leftarrow 1$
12:        Update the value of the overtime penalty $\alpha$
13:    else
14:        $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q_{max}$ was reached
15:    end if
16:    if $\hat{\mathcal{S}}'$ is a new best solution then
17:        Optional: apply the US procedure on each route of $\hat{\mathcal{S}}'$
18:        $\hat{\mathcal{S}}_{best} \leftarrow \hat{\mathcal{S}}'$
19:    end if
20:    Update the roulette wheel
21: end while
22: Apply post-optimization on $\hat{\mathcal{S}}_{best}$
```

Note that even if $\hat{\mathcal{S}}'$ is a new best R-MTVRP solution, it may be rejected. This situation only arises if $\hat{\mathcal{S}}'$ is an infeasible MTVRP solution, due to the use of an adaptive objective function (see Section 5.1.2).

5.2. Adaptive large neighborhood search for VRP combined with bin packing (ALNSP)

The ALNSP is based on two embedded loops: an outer loop and an inner loop. Let ζ be the iteration counter of the outer loop. At the initialisation step ($\zeta = 0$), a CVRP solution $\mathcal{X}^0$ is built by means of a greedy heuristic described in Section 5.3.2. An ALNS heuristic is applied to generate new CVRP solutions by iteratively destroying and repairing $\mathcal{X}^0$ (inner loop). An adaptive memory of routes (AMR) is initialized with some of the routes generated during the ALNS. At each following iteration of the outer loop ($\zeta \geq 1$), a CVRP solution $\mathcal{X}^\zeta$ is created from the routes of the AMR. Then, in the inner loop, the ALNS iteratively modifies $\mathcal{X}^\zeta$ and the AMR is updated with new routes. The BP procedure is crucial for choosing the routes that are stored in the AMR, as explained in Section 5.2.4.

5.2.1. Objective function and acceptance criterion

Even though the objective function of the MTVRP is to minimize the total travel time, the ALNSM algorithm uses a modified objective function with penalized overtime in order to guide the search process towards good R-MTVRP solutions. In the ALNSP, however, only CVRP solutions are created using ALNS iterations. Their quality is measured in terms of total travel time. After an assignment has been made to create an R-MTVRP solution, an overtime measure can be obtained.

In this work, we consider two versions of the ALNSP that differ in terms of the acceptance criterion of new solutions created by ALNS iterations within the inner loop:

- In the first version, ALNSP(a), the acceptance criterion of new CVRP solutions is solely based on the CVRP objective function (i.e., total travel time). If the quality of a candidate CVRP solution $\mathcal{X}'$ passes the acceptance criterion, then the packing procedure is applied on $\mathcal{X}'$ in order to obtain the overtime measure of a corresponding R-MTVRP solution $\hat{\mathcal{S}}'$, and the AMR is updated accordingly.

- In the second version, ALNSP(b), the acceptance criterion of a new CVRP solution $\mathcal{X}'$ is based on the quality of the R-MTVRP solution $\hat{\mathcal{S}}'$ obtained by applying the assignment procedure on $\mathcal{X}'$ before the evaluation of the acceptance function. In order to reduce the computational effort, CVRP solutions that are unlikely to yield good R-MTVRP solutions are discarded. The acceptance of a new CVRP solution $\mathcal{X}'$ is determined in two steps:

  1. Let $\psi \geq 1$ be a parameter called pre-acceptance factor. If the travel time $D_{\mathcal{X}'}$ of the candidate solution is not excessively deteriorated compared with that of the incumbent $D_{\mathcal{X}}$, i.e., if $D_{\mathcal{X}'} \leq \psi D_{\mathcal{X}}$, then a packing procedure produces an R-MTVRP solution $\hat{\mathcal{S}}'$ using the routes in $\mathcal{X}'$, and $O_{\hat{\mathcal{S}}'}$ can be calculated. Else, $\mathcal{X}'$ is automatically discarded without trying to evaluate the acceptance criterion.
  2. If $O_{\hat{\mathcal{S}}'}$ is calculated, then the candidate solution $\mathcal{X}'$ may be accepted or not based on the value of its corresponding R-MTVRP solution $\hat{\mathcal{S}}'$. The penalty associated with overtime in the objective function is the adaptive parameter α regulated exactly as in the ALNSM but reinitialized at each iteration of the outer loop.

In both versions of the ALNSP, a simulated annealing framework is used to accept or reject candidate solutions. This framework is identical to that described for the ALNSM.

In order to detect improving or new best solutions within the inner loop and to accordingly update the roulette wheel, only those solutions produced during the current iteration of the outer loop are taken into account. In the ALNSP(a), the total travel times of CVRP solutions are compared. In the ALNSP(b), the comparison of R-MTVRP solutions is performed exactly as in the ALNSM. During the course of the ALNSP, a global best R-MTVRP solution $\hat{\mathcal{S}}_{best}$ is maintained by using the same criterion as for the ALNSM. Indeed, nothing guarantees that the best R-MTVRP solution $\hat{\mathcal{S}}^{\zeta}_{best}$ found during a given iteration of the outer loop outperforms its predecessor $\hat{\mathcal{S}}^{\zeta-1}_{best}$.

5.2.2. Destroy-and-repair

The heuristics used to destroy and repair CVRP solutions are described in Section 5.3. The principle of these heuristics is exactly the same as for the ALNSM, except that the specific multi-trip operators are not applied. The number q of customers that are removed at each iteration evolves as in the ALNSM but its value is reset to 1 at each iteration of the outer loop. The roulette wheel mechanism works as explained in Section 5.1.5 but is reinitialized at each iteration of the outer loop.

5.2.3. Packing procedure

We apply the packing procedure of Olivera and Viera (2007) to assign the routes of a CVRP solution to m available vehicles in order to produce an R-MTVRP solution. This allows us to associate an overtime measure to a CVRP solution. First, a greedy heuristic iteratively assigns the longest route to the emptiest vehicle. If no feasible assignment is produced, then a labeling algorithm repeatedly attempts to reduce the overtime of the tour having the longest duration.

5.2.4. Updating the AMR

The AMR is mostly based on principles proposed in Olivera and Viera (2007). It is a list of routes, say $\mathcal{L}_M$, created during the course of the algorithm. Whenever a new R-MTVRP solution $\hat{\mathcal{S}}$ is created by the packing procedure, $\mathcal{L}_M$ is updated by inserting the routes of $\hat{\mathcal{S}}$ in the list. The routes in $\mathcal{L}_M$ are sorted in increasing order of the cost of the R-MTVRP solution to which they belong: routes belonging to "good" R-MTVRP solutions appear at the early positions of the memory. $C^M_{\hat{\mathcal{S}}} = \sum_k D_k + \alpha_M \sum_k O_k$ is the modified cost function used to assess the quality of $\hat{\mathcal{S}}$ for the purpose of inserting its routes into $\mathcal{L}_M$. Unlike α, the value of α_M is constant during the course of the algorithm.

Let $R_{\hat{\mathcal{S}}}$ be the set of routes of solution $\hat{\mathcal{S}}$. Since different R-MTVRP solutions may contain identical routes, it is frequent that a route r of $R_{\hat{\mathcal{S}}}$ is already stored in $\mathcal{L}_M$, with index i_r, before the update. In that case, if $C^M_{\hat{\mathcal{S}}}$ is less than the cost associated to the duplicate of r in $\mathcal{L}_M$, r is relocated accordingly at a position $i'_r < i_r$. The maximum size λ_M of $\mathcal{L}_M$ is determined during the configuration phase. If the size of $\mathcal{L}_M$ exceeds λ_M after an update, the routes with the highest positions are deleted until the size of $\mathcal{L}_M$ becomes equal to λ_M.

5.2.5. Using the AMR to build the inner loop's initial solution

At the beginning of each iteration ζ of the outer loop, a CVRP solution $\mathcal{X}^{\zeta}$ is created and used as an initial solution for the inner loop. This is done by using the stored routes of $\mathcal{L}_M$ when $\zeta > 0$.

Algorithm 2 Building a (partial) VRP solution from the AMR

```
Build a temporary copy $\mathcal{L}$ of $\mathcal{L}_M$: $\mathcal{L} \leftarrow \mathcal{L}_M$
$\mathcal{X}^\zeta = \emptyset$
while $\mathcal{L} \neq \emptyset$ do
    Choose one element $r$ of $\mathcal{L}$ randomly
    Add $r$ to the set of routes in $\mathcal{X}^\zeta$
    Remove from $\mathcal{L}$ any route having one or more customers in common with $r$
end while
```

Whenever a route r is added in $\mathcal{X}^\zeta$, the selection is performed randomly so that the routes that occupy first positions in $\mathcal{L}$ have a higher probability of being chosen. At each iteration of the above procedure, the selected route is the one at position $\lfloor \nu^{y_M} \times |\mathcal{L}| \rfloor$, where $\nu \sim \mathcal{U}[0, 1[$ is a random number and y_M is a parameter greater than 1. If customers remain unserved after the above procedure, then $\mathcal{X}^\zeta$ is a partial CVRP solution and the unserved customers have to be assigned. For this purpose, two options are considered and the configuration phase determines which one is applied:

- Insert unrouted customers directly into $\mathcal{X}^\zeta$ using the greedy heuristic described in Section 5.3.2 without multi-trip operators.
- Build a separate set of routes $\mathcal{R}_{unrouted}$ with the remaining customers using the same greedy heuristic as above and then set $\mathcal{X}^\zeta \leftarrow \mathcal{X}^\zeta \cup \mathcal{R}_{unrouted}$. The creation of $\mathcal{R}_{unrouted}$ keeps the routes selected from $\mathcal{L}_M$ unchanged.

5.2.6. US local search

As in the ALNSM method, the US procedure of the GENIUS algorithm is optionally applied whenever a new global best solution is found or whenever a solution is accepted.

5.2.7. Stopping criterion

The idea here is exactly the same as in the ALNSM except that the maximum number of iterations $Iter$ is treated globally: this limit is defined as the number of iterations of the outer loop multiplied by the number of iterations of the inner loop. The latter is determined by a parameter $ALNS_{iter}$. As for the ALNSM, if feasibility is not achieved after a given global number of iterations, the ALNSP continues until a feasible solution is found or until a time limit is reached.

5.2.8. Post-optimization

Once the outer loop of the ALNSP terminates, a post-optimization procedure tentatively improves the best R-MTVRP solution found. This procedure is a VND which considers VRP moves based on the routes of the R-MTVRP solution. Neighborhoods are explored in the following order: 2-opt, relocate, exchange, and US. The 2-opt neighborhood treats each route separately, meaning that the edges considered for deletion and reinsertion belong to the same route. The relocate and the exchange neighborhoods both treat a maximum chain length of four nodes, exactly as in the ALNSM. However, instead of considering the multi-trip version of the operators, each chain to be relocated or exchanged contains customers of a single route. Note that, as in the ALNSM algorithm, the relocate and the exchange operators consider moves within the same vehicle or involving two different vehicles. The US procedure is applied separately on each route that was modified since the last call of the procedure. As with the other neighborhoods, as soon as an improvement is made for one of the routes, the VND goes back to the 2-opt neighborhood. If the post-optimized R-MTVRP solution still contains overtime after the VND is applied, then the packing procedure is called in the hope of achieving feasibility or at least reducing overtime. However, if a feasible MTVRP solution was found, the packing procedure is not called since the assignment of routes to vehicles does not impact the total travel time.

5.2.9. Summary of the two versions of the ALNSP heuristic

Both versions of ALNSP are summarized in Algorithms 3 and 4

Algorithm 3 ALNSP(a): acceptance based on total duration

```
1: $\zeta \leftarrow 0$
2: Create $\mathcal{X}^0$ and apply the assignment heuristic to obtain $\hat{\mathcal{S}}_{best}$
3: **while** Stopping criterion is not met **do** *(Outer loop)*
4:     Build a new CVRP solution $\mathcal{X}^\zeta$ (out of the AMR for $\zeta > 0$)
5:     $\mathcal{X} \leftarrow \mathcal{X}^\zeta$
6:     $q = 1$; initialize the roulette wheel and the adaptive parameters for this iteration
7:     Initialize the inner loop's iteration counter: $iter_{inner} \leftarrow 0$
8:     **while** $iter_{inner} < ALNS_{iter}$ **do** *(Inner loop)*
9:         Roulette wheel: select a removal heuristic $h_{rem}$ and an insertion heuristic $h_{ins}$
10:        Remove customers from $\mathcal{X}$ using $h_{rem}$, creating a partial CVRP solution.
11:        Insert customers into the partial solution using $h_{ins}$, creating a CVRP solution $\mathcal{X}'$
12:        **if** CVRP solution $\mathcal{X}'$ meets the acceptance criterion **then**
13:            *Optional: apply the US procedure on each route of $\mathcal{X}'$*
14:            $\mathcal{X} \leftarrow \mathcal{X}'$
15:            Apply the assignment heuristic on $\mathcal{X}$ and obtain $\hat{\mathcal{S}}$
16:            Update the AMR with the routes of $\hat{\mathcal{S}}$
17:            $q \leftarrow 1$
18:            **if** $\hat{\mathcal{S}}$ is a new global best solution **then**
19:                *Optional: apply the US procedure on each route of $\mathcal{X}$*
20:                Update $\hat{\mathcal{S}}_{best}$
21:            **end if**
22:            Update the value of the overtime penalty $\alpha$
23:        **else**
24:            $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q_{max}$ was reached
25:        **end if**
26:        Update the roulette wheel
27:        $iter_{inner} \leftarrow iter_{inner} + 1$
28:    **end while**
29:    $\zeta = \zeta + 1$
30: **end while**
31: Apply post-optimization on $\hat{\mathcal{S}}_{best}$
```

Algorithm 4 ALNSP(b): acceptance based on total duration and overtime

```
1: $\zeta \leftarrow 0$
2: Create $\mathcal{X}^0$ and apply assignment heuristic to obtain $\hat{\mathcal{S}}_{best}$
3: while Stopping criterion is not met do (Outer loop)
4:     Build a new CVRP solution $\mathcal{X}^\zeta$ (out of the AMR for $\zeta > 0$)
5:     $\mathcal{X} \leftarrow \mathcal{X}^\zeta$
6:     Apply the assignment heuristic on $\mathcal{X}$ and obtain $\hat{\mathcal{S}}$
7:     $q = 1$; initialize the roulette wheel and the adaptive parameters for this iteration
8:     Initialize the inner loop's iteration counter: $iter_{inner} \leftarrow 0$
9:     while $iter_{inner} < ALNS_{iter}$ do (Inner loop)
10:        Roulette wheel: select a removal heuristic $h_{rem}$ and an insertion heuristic $h_{ins}$
11:        Remove customers from $\mathcal{X}$ using $h_{rem}$, creating partial CVRP solution
12:        Insert customers into the partial solution using $h_{ins}$, creating a CVRP solution $\mathcal{X}'$
13:        if $D_{\mathcal{X}'} < \psi D_{\mathcal{X}}$ then
14:            Apply assignment heuristic on $\mathcal{X}'$ and obtain $\hat{\mathcal{S}}'$
15:            if R-MTVRP solution $\hat{\mathcal{S}}'$ meets the acceptance criterion then
16:                Optional: apply the US procedure on each route of $\mathcal{X}'$
17:                $\mathcal{X} \leftarrow \mathcal{X}'$
18:                Update the AMR with the routes of $\hat{\mathcal{S}}'$
19:                $q \leftarrow 1$
20:                Update the value of the overtime penalty $\alpha$
21:            else
22:                $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q_{max}$ was reached
23:            end if
24:            if $\mathcal{X}$ is a new global best solution then
25:                Optional: apply the US procedure on each route of $\mathcal{X}$
26:                Update $\hat{\mathcal{S}}_{best}$
27:            end if
28:        else
29:            $q \leftarrow q + 1$, or $q \leftarrow q_{low}$ if $q_{max}$ was reached
30:        end if
31:        Update the roulette wheel
32:        $iter_{inner} \leftarrow iter_{inner} + 1$
33:    end while
34:    $\zeta = \zeta + 1$
35: end while
36: Apply post-optimization on $\hat{\mathcal{S}}_{best}$
```

5.3. Removal and insertion heuristics

The ALNSM and the ALNSP algorithms use insertion and removal heuristics at each iteration in order to destroy and repair solutions. In the case of the ALNSM, the solutions treated by means of these heuristics are R-MTVRP solutions where overtime is allowed. In the ALNSP context, these heuristics work on CVRP solutions.

5.3.1. Removal heuristics

At each iteration of the ALNSM or of the ALNSP inner loop, a removal heuristic is randomly chosen by the roulette wheel mechanism. All removal heuristics randomly select customers to be removed. In most of these heuristics, randomization is applied based on a list of customers, denoted L, sorted according to the criterion of the considered heuristic. The customers with a small index in the sorted list are more likely to be removed. As proposed in Ropke and Pisinger (2006a), each time a customer is to be removed, an index *rand* is determined as $\lfloor \nu^{y_{rem}} \times |L| \rfloor$, where $\nu \sim \mathcal{U}[0, 1[$. The customer with index *rand* is removed and L is updated. The same parameter $y_{rem} \geq 1$ controls the randomization level of the worst removal heuristics, the Shaw removal heuristic, the historical removal heuristics, and the cloud removal heuristic.

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

Worst removal. We adapt the worst removal heuristic of Ropke and Pisinger (2006a), employing different cost measures. Let i be a customer of an R-MTVRP solution $\hat{\mathcal{S}}$ or a CVRP solution $\mathcal{X}$ and D_k^i (resp., O_k^i) be the total travel time (resp., overtime if applicable) of the vehicle serving i in $\hat{\mathcal{S}}$ or $\mathcal{X}$. Let D_k^{i*} (resp., O_k^{i*}) be the total travel time (resp., overtime) of the same vehicle if i is removed. Finally, let c_i be the cost measure associated to i. In the different versions of the heuristic, c_i is defined as *a)* $D_k^i - D_k^{i*}$; *b)* $D_k^i - D_k^{i*} + \alpha(O_k^i - O_k^{i*})$, where α is the overtime penalty for the current iteration (see Section 5.1.2); *c)* $(D_k^i - D_k^{i*})/\bar{c_i}$, where $\bar{c_i}$ is the average cost of customer i as defined in a over all preceding iterations. Each cost measure gives rise to a different version of the heuristic. Of course, only a and c are applicable in the case of the ANLSP since partial CVRP solutions do not contain the notion of overtime.

Shaw removal. The idea of removing customers based on their similarities was proposed by Shaw (1998) and extended by Ropke and Pisinger (2006a). The first customer is chosen randomly and the remaining $q-1$ customers are chosen based on their relatedness with the first one. In this work, the relatedness measure of a customer i with respect to customer j takes into account the coordinates and the demands of both customers and is defined as $\Phi(t_{ij}/t_{max}) + (1-\Phi)(|d_i - d_j|/d_{max})$, where t_{max} is the maximum travel time between any pair of customers in G, d_{max} is the maximum demand, and $\Phi \in [0,1]$ is a parameter that determines the relative weights of the time and demand factors.

Historical removal. We exploit the historical information captured in previous iterations within two removal heuristics: the route-based historical removal heuristic and the vehicle-based historical removal heuristic. These heuristics are based on the Shaw removal heuristic with a relatedness measure that encompasses historical information about pairs of customers. The principle is similar to that of the request graph removal heuristic of Ropke and Pisinger (2006b), except that the weight h_{ij} of an edge (i,j) of the request graph is the average cost of the λ_H best solutions found so far with i and j placed in the same route (resp. vehicle). The Shaw removal heuristic is applied, using $1/h_{ij}$ as the relatedness measure of customers i and j. Whenever a new solution is visited, the historical memory is updated for each couple (i,j) placed on the same route (resp., vehicle). For the purpose of both heuristics, the solution values are measured in terms of the sum of duration and penalized overtime. The overtime penalization factor α_H is a parameter whose value is fixed to avoid inconsistencies between the solution costs stored within the historical memory. In the ALNSP algorithm, only the route-based historical removal is used since the two heuristics are equivalent when each vehicle performs a single route.

Cloud removal. The cloud removal heuristic tends to remove a "cloud" of customers that are geographically close to one another, starting from a random customer i. In this case, the customers in L are sorted in increasing order of their distance to the center of gravity of the set of customers already chosen to be removed. To compute this center of gravity, only the coordinates of the customers are taken into account, not their demands, i.e., every customer to be removed is given the same weight.

Route removal. In the route removal heuristic, one of the routes is completely removed, i.e., all its customers are removed along with one of its internal depots if any. The removed route is the one having the smallest $|q - n_r|$ value, where n_r is the number of customers of route r. Ties are broken arbitrarily.

Merging routes. Merging two routes consists of removing the internal depot that separates their respective customer sequences. In our ALNSM implementation, two versions of each removal heuristic are considered: with and without merging routes. In both versions, customers are removed following the rules of the removal heuristic under consideration. When empty routes are created, one of their internal depots is also removed, as explained in Section 4.1.2. In the version that includes the merge operator, the remaining routes are merged recursively if possible starting from the left-most route of each tour. That is, we consider the removal of each internal depot, in the order they appear in the tour sequence. If the depot removal is feasible with respect to capacity constraints, then the merge is performed. The merge operator is included by default in the removal heuristics. A single parameter determines whether the heuristic versions without merge are also included in the ALNSM.

5.3.2. 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 $\mathcal{X}$ until all customers have been routed. When inserting a customer in $\hat{\mathcal{S}}$, two design choices are considered during the configuration phase: the insertion cost is either the increase in travel time, or the increase in travel time, plus the penalized increase in overtime. In the latter case, the associated penalty factor α is the same as the one used for the acceptance criterion. In the "repair" phase of ALNSM, the four insertion schemes described in Section 4.1 are embedded in insertion heuristics detailed below. When repairing a partial solution $\hat{\mathcal{S}}$ by inserting unrouted customers, overtime is allowed. Since the insertion costs are solely based on time related considerations, if a customer can be inserted with scheme 1 without violating the capacity constraints, then schemes 2, 3 and 4 do not need to be considered as long as the triangular inequality holds. Similarly, scheme 4 needs to be considered only if any other scheme is infeasible. In the ALNSP context, only scheme 1, which is the classical VRP insertion scheme, is used.

Greedy heuristic. The greedy heuristic iteratively adds a customer at its best possible position in the current partial solution $\hat{\mathcal{S}}$ or $\mathcal{X}$. The customer is chosen so as to minimize the cost increment of the solution.

Route-based regret heuristic. In the context of VRP, it is common to employ regret heuristics in the construction phase (e.g. Tillman and Cain, 1972; Potvin and Rousseau, 1993; Diana and Dessouky, 2004; Ropke and Pisinger, 2006a,b). A simple approach is to define the regret as the slack between the cost of inserting a customer in its best possible route and the cost of inserting the same customer in its second best route. Let $\Delta C^R_{j,y}$ be the variation in the value of the cost function of $\hat{\mathcal{S}}$ or $\mathcal{X}$ if customer j is inserted in the best possible position in the route where its insertion cost is the y^{th} lowest. At each iteration of the route-based regret heuristic, a customer i is selected as follows: $i = \text{argmax}_{j\in U}\{\Delta C^R_{j,2} - \Delta C^R_{j,1}\}$, where U is the set of unrouted customers. Customer i is then inserted in its minimum insertion cost position. The regret-x heuristic generalizes this definition by considering all positions until the x^{th} one: a customer i is selected as $\text{argmax}_{j\in U}\sum_{y=2,\ldots,x}\{\Delta C^R_{j,y} - \Delta C^R_{j,1}\}$.

Vehicle-based regret heuristic. The principle of this heuristic is the same as above except that vehicles are considered as insertion choices, instead of routes. At each iteration, select customer $i = \text{argmax}_{j\in C}\sum_{y=2,\ldots,x}\{\Delta C^V_{j,y} - \Delta C^V_{j,1}\}$, where $\Delta C^V_{j,y}$ is the variation in the value of the objective function if customer j is inserted at its best position in the vehicle with the y^{th} lowest insertion cost.

Position-based regret heuristic. At each iteration of the position-based regret-x heuristic, a customer i is selected such that $i = \text{argmax}_{j\in U}\sum_{y=2,\ldots,x}\{\Delta C^P_{j,y} - \Delta C^P_{j,1}\}$, where $\Delta C^P_{j,y}$ is the variation in the value of the objective function if customer j is inserted in the position with the y^{th} lowest insertion cost. Again, the customer insertion is performed at minimum cost.

Insertion heuristics listing. Both route-based regret heuristic and position-based regret heuristic can be applied in ALNSM and ALNSP. So can the greedy heuristic. However, the vehicle-based regret heuristic is only used in the ALNSM. Indeed, since the insertion heuristics of the ALNSP work on partial CVRP solutions, this implies that each vehicle performs a single route and thus routes and vehicles are equivalent. Whenever one of the regret heuristics is applied, the value of x must be specified. In this work, a subset of the possible values of x is associated with each version of the regret heuristic. When the roulette wheel randomly selects an insertion heuristic, the following choices are taken into consideration:

Heuristic	Possible values of x	Applied in ...	Implementation
Greedy	-	ALNSM, ALNSP	by default
Route-based regret	$2, 3, 4, 5$	ALNSM, ALNSP	optional
Vehicle-based regret	$2, ..., \min\{4, m-1\}$	ALNSM	optional
Position-based regret	$2, 3, \lfloor q/2 \rfloor, q$	ALNSM, ALNSP	optional

For example, for the position-based insertion heuristic, four values of x are considered, and each gives rise to a different candidate heuristic for the roulette wheel selection. The set of candidate insertion heuristics available for selection by the roulette wheel in the ALNSM or ALNSP is determined in the configuration phase. The greedy heuristic is applied by default, in conjunction with up to three of the above groups of regret heuristics for the ALNSM, and up to two in the case of the ALNSP.

Tabu edges. The tabu-edge mechanism is an optional feature that affects the insertion process. The parametrization phase determines whether it is used or not. The goal of this optional feature is to avoid inserting edges that have been removed too often during previous ALNS iterations. If the tabu-edge mechanism is activated, the cost of edges are modified when evaluating the insertion costs. For each edge (i, j), a penalty $p_{i,j}$ is initialized at 0. Each time a removal heuristic is called, the penalties are updated as follows:

- if edge (i, j) was not removed during the current ALNS iteration, $p_{i,j} = p_{i,j} \times \gamma$,
- if edge (i, j) was removed during the current ALNS iteration, $p_{i,j} = p_{i,j} \times \gamma + \Gamma$.

The parameter $\gamma \in [0, 1]$ represents the persistence of the penalty through ALNS iterations and Γ is the penalization of the edges that are removed during the current ALNS iteration. Let $c_{i,j}$ be the real cost of edge (i, j). If the tabu-edge mechanism is activated, the modified cost of an edge during the insertion phase is $c^{modified}_{i,j} = c_{i,j} \times (1 + p_{i,j})$.

6. Computational experiments

We have performed extensive computational experiments to assess the performance of our heuristics and to analyze the behavior of several algorithmic components.

6.1. Benchmark instances

We provide results obtained with both ALNSM and ALNSP methods for the 104 benchmark instances generated by Taillard et al. (1996). Each of these is made up of three components: a CVRP instance, a fleet size m, and $T_{max} = [\tau z^*/m]$, where z^* is the total duration of the best CVRP solution obtained by the tabu search heuristic of Rochat and Taillard (1995). The smaller the value of τ, the greater the difficulty to achieve feasibility. Two tour durations were considered to create the benchmark instances: $T_1 = [1.05z^*/m]$ and $T_2 = [1.10z^*/m]$. The CVRP instances used to create the MTVRP benchmark instances are the instances CMT-1 to CMT-5, CMT-11 and CMT-12 of Christofides et al. (1979), and the instances F-11 and F-12 of Fisher (1994). Combining these CVRP instances with different values of m as shown in Table 2, and with the two values of T_{max}, yields our 104 benchmark instances. In the following, all MTVRP instances created from the same CVRP instance are said to form a class of instances.

CVRP instance (customers)	Values of m	z^*
CMT-1 (50)	1,...,4	524.61
CMT-2 (75)	1,...,7	835.26
CMT-3 (100)	1,...,6	826.14
CMT-4 (150)	1,...,8	1028.42
CMT-5 (199)	1,...,10	1291.44
CMT-11 (120)	1,...,5	1042.11
CMT-12 (100)	1,...,6	819.56
F-11 (71, clustered)	1,...,3	241.97
F-12 (134, clustered)	1,...,3	1162.96

Table 2: Characteristics of the MTVRP benchmark instances

6.2. Automatic configuration

A configuration is a set of values assigned to the algorithmic parameters. The configuration of our ALNSM and ALNSP algorithms is performed by using the `irace` package (López-Ibáñez et al., 2011), an automatic configuration tool. For each algorithm, a set of parameters whose values need to be determined is provided to `irace` along with their allowed range. We also supply `irace` with a set of MTVRP instances, called training instances. We determine the training budget, i.e., the total number of configurations that should be tested. The automatic configuration tool `irace` relies on an iterative process in which algorithm configurations compete with each other. In order to configure an algorithm, i.e., assigning a value to each algorithmic parameter, `irace` initially generates several configurations. By comparing the performances of the generated configurations on training instances, the configuration tool chooses those that will be eliminated and those that deserve further investigation. Those configurations that are not eliminated during a given iteration of the configuration process serve as a basis to generate more configurations in the next iteration. When the total number of examined configurations hits the training budget, the automatic configuration tool yields a set of configurations that are statistically equally well performing on the training instances. By default, `irace` returns seven configurations, that we call "final configurations" for each algorithm.

To avoid biasing the results, the configuration of both algorithms is performed on a set of training instances which contains none of the benchmark instances. Previous authors of MTVRP heuristics have configured their algorithm on a small subset of benchmark instances. Working with as few as six or seven instances out of the 104 available may seem insignificant, but in fact, many MTVRP instances share exactly the same underlying CVRP data. We have observed that using training instances instead of a subset of benchmark instances for configuration purposes mainly results in a slight decrease of the average quality of solutions measured over five runs. This suggests that the results we present in Section 6.3 could be improved mainly in terms of robustness if a subset of benchmark instances was used as the set of training instances during the configuration phase. We have constructed five CVRP instances whose distributions of customer demands and coordinates simulate the CVRP instances of Christofides et al. (1979). We also used two clustered CVRP instances from Taillard (1993), since they share common characteristics with those of Fisher (1994). We have run our ALNSM algorithm with a manual configuration to produce a z^* value for each of these seven CVRP instances. We set $\tau = 1.05$ and chose different values of m to create a large number of MTVRP instances. We then ran the ALNSM and ALNSP algorithms with manual configurations on these MTVRP instances in order to identify those for which the algorithms usually fail to obtain a feasible solution. The training instances for `irace` were then mostly (but not only) chosen as those that seemed difficult to solve for our ALNSM and ALNSP.

A cost function must be provided in order to optimize an algorithm configuration with `irace`. In the case of the MTVRP, using the objective function of the problem (i.e., the total travel time) is not possible because infeasible solutions may have a small travel time. This is why the objective function of the MTVRP considered during the configuration phase was the travel time penalized with the total overtime of the solution, using a very large constant for the penalty factor. This way, the focus is first set on feasibility. Also, since the running time of the algorithms may vary significantly for a given instance, depending on whether and when a feasible solution is found, `irace` was used to provide acceptable "anytime configurations". These configurations can be outperformed by other sets of parameters for a given running time, but they are assumed to provide acceptably good quality results for any running time. The training budget for each of the two considered algorithms was set to 80 000. The training phase was 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). Due to the high degree of parallelization of our tests (up to 80 cores), a training only takes a few hours to find the statistically best configurations.

Not only does the configuration tool determine the value of numerical parameters, but it also selects algorithmic design options. For example, the cloud removal heuristic becomes available for selection through the roulette wheel only if the corresponding Boolean parameter takes the value *True* in the configuration given by `irace`. In the following, we divide the parameters into two categories for the sake of convenience: the Boolean parameters which represent a design choice between two algorithmic options, and the numerical parameters, which take continuous or discrete values contained in a given interval.

Table 3 provides a complete list of design choices. The leftmost column contains a classification of the parameters. Each of the design choices can be set to *True* or *False* during the configuration phase. For both ALNS methods, the value taken by the design choices in the seven final configurations is reported in the rightmost columns. All the design choices took the same value for all the final configurations of the ALNSP and of the ALNSM respectively, except for the use of the worst removal heuristic (variant b) in the ALNSM that was six times *True* and one time *False* (see the asterisk in Table 3). In Table 4, the type and the range of all numerical parameters (either real or integer) are specified. Some parameters are mandatory, irrespective of the implementation choices, while others must be configured only for certain choices. We provide summarized data on the value taken by each numerical parameter in the final ALNSM and ALNSP configurations.

	Design choice	ALNSM	ALNSP
Removal heuristics	Use random removal.	F	F
	Use worst removal (variant a).	T	T
	Use worst removal (variant b).	T^*	-
	Use worst removal (variant c).	F	T
	Use Shaw removal.	T	T
	Use route removal.	F	F
	Use cloud removal.	F	F
	Use historical removal (route-based information).	F	-
	Use historical removal (vehicle-based information).	F	F
	Include removal heuristics without the merge operator.	T	-
Insertion heuristics	Use position-based regret insertion.	F	F
	Use route-based regret insertion.	T	T
	Use vehicle-based regret insertion.	F	-
	Use insertion costs with penalized overtime.	T	-
	Use the tabu-edge mechanism.	F	F
Local search	Use US on each new best solution.	F	F
	Use US after a solution is accepted.	F	F
ALNSP (a) or (b)	Accept a solution based on its penalized cost (T = ALNSP(b)).	-	T
ALNSP CVRP solutions	When building a CVRP solution from the AMR, route the remaining customers separately.	-	T

Table 3: Boolean parameters: design choices in the final configurations of ALNSM and ALNSP

				ALNSM			ALNSP		
	Name	Type	Authorized range	Final range	Average	Chosen configuration	Final range	Average	Chosen configuration
Objective function (5.1.2, 5.2.1)	α_{min}	integer, step 10	[0,100]	[40,70]	60	70	[0,30]	10	30
	α_{max}	integer, step 10	[α_{min},10000]	[2900,7050]	4750	6610	[2890,6760]	4540	3770
	μ	real	[1,2]	[1.2131,1.7034]	1.425	1.5701	[1.333,1.9623]	1.6961	1.7446
	ξ	integer, step 100	[100,1000]	[100,100]	100	100	[100,900]	400	300
Simulated annealing (5.1.3)	θ_0	real	[0.01,0.1]	[0.07,0.0987]	0.0875	0.0709	[0.0703,0.0956]	0.0830	0.0876
	κ	real	[0,1]	[0.7995,0.9953]	0.9200	0.9953	[0.1742,0.9484]	0.5933	0.6274
	η	real	[0.9,1]	[0.9373,0.9968]	0.9802	0.9373	[0.9599,0.9831]	0.9686	0.9781
Roulette wheel (5.1.5, 5.2.2)	σ_1	real	[0,1]	[0.1586,0.4777]	0.3145	0.4162	[0.0521,0.2675]	0.0971	0.0705
	σ_2	real	[0,1]	[0.0321,0.4196]	0.2977	0.2657	[0.1586,0.4777]	0.3145	0.2214
	ρ	real	[0,1]	[0.6796,0.9457]	0.8034	0.7158	[0.0072,0.934]	0.5621	0.7685
	Θ	integer, step 100	[100,5000]	[1000,4700]	3400	4000	[500,4900]	2700	3000
Bounds on q (5.1.4)	υ	real, 2 digits	[0.05,0.40]	[0.34,0.39]	0.36	0.34	[0.25,0.40]	0.34	0.39
	δ	integer	[5,10]	[7,9]	8	9	[6,10]	8	9
Randomization factor (5.3.1)	p	real	[1.10,10]	[6.1684,8.0679]	6.968	7.2345	[3.1702,8.7392]	6.9649	8.2864
Shaw removal (5.3.1)	Φ	real	[0,1]	[0.7806,0.9916]	0.8934	0.9916	[0.7869,0.9631]	0.8593	0.8183
Historical removal (5.3.1)	λ_H	integer, step 5	[5,100]	-	-	-	-	-	-
	α_H	integer, step 10	[10,10000]	-	-	-	-	-	-
Tabu-edge mechanism (5.3.2)	γ	real	[0,1]	-	-	-	-	-	-
	Γ	real	[0,10]	-	-	-	-	-	-
AMR (5.2.4, 5.2.5)	λ_M	integer, step 10	[10,500]	-	-	-	[280,460]	360	340
	y_M	real	[1,10]	-	-	-	[2.236,5.3719]	3.9841	4.4967
	α_M	integer	[0,100000], step 10	-	-	-	[47520,76150]	62960	76150
ALNSP(b) pre-acceptance factor (5.2.1)	ψ	real	[1,1.10]	-	-	-	[1.0313,1.0947]	1.0714	1.0804
ALNSP inner loop (5.2)	$ALNS_{iter}$	integer, step 100	[100,50000]	-	-	-	[8900,19600]	12500	9200
US (also for post-opt.)	p_{US}	integer	[3,10]	[4,6]	5	6	[5,8]	6	6

Table 4: Numerical parameters in the final configurations of ALNSM and ALNSP

In Section 6.3, we present the results obtained on benchmark instances with one ALNSM configuration and one ALNSP configuration. The chosen configurations are those that yield the best results in terms of feasibility on the training instances for each algorithm. In the following sections, when we mention the results of the ALNSM and ALNSP algorithms, we refer to these two specific configurations. The associated parameter values are provided in Tables 3 and 4.

6.3. Numerical results

Detailed numerical results were obtained for the ALNSM and ALNSP algorithms by running both configurations five times on each of the 104 benchmark instances. The tests were performed on a computer with 3.20 Ghz processor Intel(R) Core(TM) i7-3930K with 64 Gigabytes of RAM under Windows 7. Note that the amount of RAM used for a single run is negligible.

The default stopping criterion for each run is to stop after 250 000 iterations if a feasible solution is found. In the case of the ALNSP, this criterion is met when $\zeta \times ALNS_{iter}$ exceeds 250 000. If no feasible solution is found after 250 000 iterations, then the algorithm runs until it finds one, or until a time limit ($n^2/50$ seconds) is reached.

We classify the benchmark instances as in Cattaruzza et al. (2014b). The instances in group 1 (G1) were solved to optimality by Mingozzi et al. (2013). The instances in group 2 (G2) were not solved to optimality but are known to be feasible. The five instances in group 3 (G3) are not known to be feasible. Note that instance CMT-4 with $m = 7$ and $T_{max} = T_1$ is classified in G3 in the paper by Cattaruzza et al. (2014b). Since a feasible solution has been found in that same paper, this instance is now classified in G2. The total number of unsolved benchmark instances (out of the complete set of 104 instances) is reported in Table 5 for our ALNSM and ALNSP algorithms and for previous authors.

Abbreviation	Reference	Unsolved
TLG	Taillard, Laporte, and Gendreau (1996)	18
BM	Brandão and Mercer (1998)	15
PS	Petch and Salhi (2004)	28
SP	Salhi and Petch (2007)	40
OV	Olivera and Viera (2007)	6
CAFV	Cattaruzza, Absi, Feillet, and Vidal (2014b) (algorithm MA)	6
CAFV+	Cattaruzza, Absi, Feillet, and Vidal (2014b) (algorithm MA+CLS)	5
ALNSM		7
ALNSP		5

Table 5: Number of unsolved benchmark instances for various algorithms

Tables 6 and 7 provide numerical results for both ALNS configurations. The best known solution value is optimal and reported in bold for instances in G1. For instances in G2, the "Previous best known" column shows the best known solution values provided by other authors when available. Bold numbers in our results indicate that we match or improve the previous best known solution values. For each configuration, we record the best, the worst, and the average value found over the runs yielding a feasible solution. These values are reported in terms of the percentage of deviation, also called "gap" in what follows, compared with the previous best known MTVRP solution value for the considered instance. For example, a gap of 1.3 gives a solution value of $1.013 \times z^*_{MTVRP}$, where z^*_{MTVRP} is the value reported in the "Optimum" or "Previous best known" column. For both methods, the number of runs yielding a feasible solution is given, as well as the number of runs yielding an optimal solution for instances in G1.

The ALNSM finds the optimal solution for 17 instances over the 42 of group G1. Optimal solution values were obtained for 33 instances with the ALNSP. Both methods produce feasible solutions for all instances in G1. Every run of the ALNSP yields a feasible solution for all instances. Concerning the instance set G2, a new best solution is found by the ALNSM for instance CMT-2 with $m = 6$ and $T_{max} = 146$. The same new best solution value is found by the ALNSP along with five other improved best known solution values. The ALNSM and the ALNSP provide exactly the previous best known solution values for three and nine instances respectively. The ALNSM yields feasible solutions for 55 instances out of 57 and the ALNSP produces feasible solutions for all instances. Table 8 provides a summary of the computing times for both ALNS configurations. For each instance, the computing time of the run providing the highest quality solution is recorded. Table 8 gives the average of these computing times for each instance class, considering only those runs that produced a feasible solution. A comparison with other authors is difficult to make due to the different stopping criteria and to the way computing times are reported. Also, data allowing a fair comparison between the CPU speeds are not in general available. Olivera and Viera (2007) provide their computing time for the best of five runs on each instance, allowing to calculate their average on each instance class. In Cattaruzza et al. (2014b), the computing

				ALNSM					ALNSP				
VRP	m	T_{max}	Optimum	# Opt	# Feas	Best gap	Worst gap	Average gap	# Opt	# Feas	Best gap	Worst gap	Average gap
CMT-1	1	551	**524.61**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	2	275	**533.00**	0	4	0.6	2.8	1.4	2	5	**0.0**	1.9	1.0
	1	577	**524.61**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	2	289	**529.85**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	4	144	**546.29**	0	1	2.3	2.3	2.3	2	5	**0.0**	1.8	1.1
CMT-2	1	877	**835.26**	0	5	0.3	0.7	0.5	0	5	0.0	0.5	0.2
	2	439	**835.26**	0	5	0.2	0.7	0.4	3	5	**0.0**	0.1	0.0
	3	292	**835.26**	0	5	0.1	0.6	0.3	1	5	**0.0**	0.6	0.1
	4	219	**835.26**	2	5	**0.0**	0.6	0.3	0	5	0.0	0.4	0.1
	5	175	**835.80**	0	5	0.9	2.5	1.7	0	5	0.0	0.1	0.1
	1	919	**835.26**	0	5	0.1	0.9	0.5	2	5	**0.0**	0.2	0.1
	2	459	**835.26**	2	5	**0.0**	0.6	0.3	0	5	0.1	0.2	0.1
	3	306	**835.26**	0	5	0.2	0.7	0.5	0	5	0.0	0.2	0.1
	4	230	**835.26**	0	5	0.0	0.8	0.4	1	5	**0.0**	0.1	0.1
	5	184	**835.26**	1	5	**0.0**	0.5	0.2	2	5	**0.0**	0.1	0.0
	6	153	**839.22**	0	5	0.2	1.2	0.7	5	5	**0.0**	0.0	0.0
CMT-3	1	867	**826.14**	0	5	0.2	0.5	0.4	2	5	**0.0**	0.3	0.1
	2	434	**826.14**	0	5	0.4	0.5	0.5	0	5	0.2	0.4	0.3
	3	289	**826.14**	0	5	0.4	0.6	0.5	1	5	**0.0**	0.4	0.3
	1	909	**826.14**	0	5	0.5	0.9	0.7	0	5	0.2	0.4	0.3
	2	454	**826.14**	0	5	0.4	0.6	0.5	1	5	**0.0**	0.4	0.3
	3	303	**826.14**	0	5	0.3	0.6	0.5	3	5	**0.0**	0.4	0.1
	4	227	**826.14**	0	5	0.5	0.9	0.7	1	5	**0.0**	0.4	0.3
CMT-11	1	1094	**1042.11**	2	5	**0.0**	0.0	0.0	2	5	**0.0**	0.4	0.2
	2	547	**1042.11**	0	5	0.4	0.6	0.4	0	5	0.1	2.9	0.9
	3	365	**1042.11**	2	5	**0.0**	0.0	0.0	3	5	**0.0**	0.7	0.2
	5	219	**1042.11**	1	5	**0.0**	0.2	0.1	3	5	**0.0**	0.3	0.1
	1	1146	**1042.11**	0	5	**0.0**	0.1	0.1	1	5	**0.0**	2.9	1.8
	2	573	**1042.11**	0	5	**0.0**	0.1	0.1	3	5	**0.0**	0.7	0.3
	3	382	**1042.11**	1	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	4	287	**1042.11**	0	5	**0.0**	0.2	0.1	5	5	**0.0**	0.0	0.0
	5	229	**1042.11**	0	5	**0.0**	0.2	0.1	5	5	**0.0**	0.0	0.0
CMT-12	1	861	**819.56**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	2	430	**819.56**	0	5	0.6	0.6	0.6	5	5	**0.0**	0.0	0.0
	3	287	**819.56**	0	5	0.6	2.7	2.1	5	5	**0.0**	0.0	0.0
	4	215	**819.56**	2	3	**0.0**	0.1	0.0	5	5	**0.0**	0.0	0.0
	1	902	**819.56**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	2	451	**819.56**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	3	301	**819.56**	5	5	**0.0**	0.0	0.0	5	5	**0.0**	0.0	0.0
	4	225	**819.56**	4	5	**0.0**	0.6	0.1	5	5	**0.0**	0.0	0.0
	5	180	**824.78**	2	5	**0.0**	0.3	0.1	5	5	**0.0**	0.0	0.0
	6	150	**823.14**	0	5	2.4	6.1	4.4	0	5	0.1	3.8	3.1
Total				***54***	***203***				***113***	***210***			

Table 6: Computational results for the instances of group G1

				ALNSM				ALNSP			
VRP	m	T_{max}	Previous best known	# Feas	Best gap	Worst gap	Average gap	# Feas	Best gap	Worst gap	Average gap
CMT-1	3	192	552.68	5	0.5	1.2	0.7	5	0.5	1.0	0.7
CMT-2	6	146	858.58	1	**-0.38**	-0.4	-0.4	4	**-0.38**	0.0	-0.2
	7	131	844.70	5	1.3	2.0	1.7	5	0.4	0.7	0.5
CMT-3	4	217	829.45	5	0.3	1.3	0.7	5	**0.0**	0.1	0.0
	5	173	832.89	3	1.7	2.3	2.0	5	**0.0**	1.5	0.5
	6	145	836.22	5	1.5	2.8	2.1	5	**0.0**	3.1	1.5
	5	182	832.34	5	0.1	0.2	0.1	5	**-0.14**	0.2	0.0
	6	151	834.35	5	0.2	0.3	0.2	5	0.0	0.2	0.1
CMT-4	1	1080	1031.00	5	0.8	1.1	0.9	5	0.6	1.1	0.9
	2	540	1031.07	5	0.4	1.2	0.8	5	0.8	1.0	0.9
	3	360	1028.42	5	0.7	1.2	1.0	5	0.2	1.2	0.8
	4	270	1031.10	5	0.9	1.4	1.1	5	0.6	1.4	1.0
	5	216	1031.07	5	1.3	3.3	2.2	5	0.6	1.2	0.8
	6	180	1034.61	5	2.4	4.0	3.5	5	0.7	2.2	1.7
	7	154	1068.59	0	-	-	-	1	**-0.11**	-0.1	-0.1
	8	135	1056.54	1	1.1	1.1	1.1	3	0.2	0.8	0.5
	1	1131	1031.07	5	1.8	2.1	2.0	5	1.2	1.9	1.6
	2	566	1030.45	5	1.5	1.8	1.6	5	1.6	1.8	1.7
	3	377	1031.59	5	1.1	1.7	1.5	5	1.1	1.4	1.2
	4	283	1031.07	5	1.0	1.5	1.3	5	0.6	1.4	1.1
	5	226	1030.86	5	1.0	1.8	1.5	5	1.0	1.6	1.4
	6	189	1030.45	5	0.9	1.9	1.6	5	1.1	1.6	1.3
	7	162	1036.08	5	1.1	2.5	1.6	5	**-0.33**	0.6	0.1
	8	141	1044.32	5	1.3	1.9	1.5	5	0.2	0.8	0.6
CMT-5	1	1356	1302.43	5	0.8	1.4	1.1	5	0.5	1.4	1.1
	2	678	1302.15	5	0.8	1.4	1.2	5	0.8	1.4	1.1
	3	452	1301.29	5	1.0	1.5	1.3	5	0.5	1.6	1.2
	4	339	1304.78	5	0.7	1.4	1.2	5	1.0	1.6	1.2
	5	271	1300.02	5	1.4	2.3	1.8	5	1.0	1.6	1.3
	6	226	1303.37	5	0.8	1.3	1.0	5	1.1	1.8	1.4
	7	194	1309.40	5	0.1	1.7	1.2	5	**-0.32**	1.0	0.6
	8	170	1303.91	5	1.1	2.2	1.7	5	**-0.06**	1.7	1.0
	9	151	1307.93	5	2.0	3.2	2.8	5	0.3	2.7	1.3
	10	136	1323.01	1	2.2	2.2	2.2	5	0.0	0.8	0.5
	1	1421	1299.86	5	2.8	3.2	3.0	5	2.7	3.2	3.0
	2	710	1305.35	5	2.1	2.7	2.5	5	2.3	2.6	2.4
	3	474	1301.03	5	2.0	2.7	2.4	5	2.2	3.1	2.6
	4	355	1303.65	5	1.4	2.0	1.7	5	2.4	3.0	2.7
	5	284	1300.62	5	1.6	2.3	2.0	5	2.4	3.1	2.7
	6	237	1306.17	5	1.7	2.2	1.9	5	1.9	2.5	2.2
	7	203	1301.54	5	2.0	2.5	2.3	5	2.5	2.6	2.6
	8	178	1308.78	5	1.2	1.7	1.4	5	0.6	1.7	1.3
	9	158	1307.25	5	1.3	1.9	1.7	5	0.6	1.5	1.1
	10	142	1308.81	5	1.4	2.4	2.0	5	0.6	1.0	0.7
CMT-11	4	274	1078.64	5	0.0	1.1	0.4	2	0.0	0.7	0.3
CMT-12	5	172	845.56	0	-	-	-	1	**0.0**	0.0	0.0
F-11	1	254	241.97	5	**0.0**	0.0	0.0	5	**0.0**	3.2	1.3
	2	127	250.85	1	**0.0**	0.0	0.0	4	**0.0**	0.0	0.0
	1	266	241.97	5	**0.0**	0.0	0.0	5	**0.0**	0.0	0.0
	2	133	241.97	5	4.8	4.8	4.8	5	**0.0**	0.0	0.0
	3	89	254.07	3	0.3	1.1	0.6	5	**0.0**	0.0	0.0
F-12	1	1221	1162.96	5	0.9	1.2	1.0	5	0.1	2.1	1.2
	2	611	1162.96	5	0.3	0.7	0.4	5	0.1	0.9	0.4
	3	407	1162.96	5	2.6	4.4	3.5	5	0.1	2.5	1.2
	1	1279	1162.96	5	0.4	0.7	0.5	5	0.0	1.0	0.5
	2	640	1162.96	5	0.0	0.4	0.3	5	0.3	1.2	0.9
	3	426	1162.96	5	2.1	2.1	2.1	5	0.1	1.1	0.8
Total				***255***				***270***			

Table 7: Computational results for the instances of group G2

times are calculated as the average of five runs for each instance and are then aggregated for each instance class. Olivera and Viera (2007) use a 1.8 GHz AMD Athlon XP2200+ CPU, while Cattaruzza et al. (2014b) work with a Intel Xeon 2.80 GHz.

	ALNSM	ALNSP	OV	CAFV	CAFV+
CMT-1 (50)	24	47	16	10	30
CMT-2 (75)	34	61	29	25	118
CMT-3 (100)	40	67	27	52	173
CMT-4 (150)	88	78	68	169	493
CMT-5 (199)	135	85	125	354	1284
CMT-11 (120)	71	68	28	99	302
CMT-12 (100)	39	59	27	37	138
F-11 (71)	56	80	13	21	40
F-12 (134)	72	71	31	87	160

Table 8: Computing times in seconds for various algorithms

The quality of the results reported in Tables 6 and 7 tends to decrease with the instance size. However, as can be seen from Table 8, computing times increase very slowly with the instance size if compared with the results reported by previous authors. This can be observed in particular for the unclustered instances CMT-4 and CMT-5. Table 9 shows the improvement in the quality of the results obtained on the largest instances when the number of iterations increases from 250 000 to 1 000 000. Due to the definition of the stopping criterion, the running time of both ALNS algorithms increases on average less than proportionally with the limit number of iterations.

VRP	m	T_{max}	Previous best known	ALNSM (250k)		ALNSM (1000k)		ALNSP (250k)		ALNSP (1000k)	
				Best gap	Avg gap	Best gap	Avg gap	Best gap	Avg gap	Best gap	Avg gap
CMT-4	1	1080.0	1031.00	0.8	0.9	0.6	0.7	0.6	0.9	0.3	0.6
	2	540.0	1031.07	0.4	0.8	0.2	0.3	0.8	0.9	0.0	0.3
	3	360.0	1028.42	0.7	1.0	0.4	0.6	0.2	0.8	0.1	0.4
	4	270.0	1031.10	0.9	1.1	0.1	0.6	0.6	1.0	0.0	0.4
	5	216.0	1031.07	1.3	2.2	0.2	1.2	0.6	0.8	**-0.06**	0.3
	6	180.0	1034.61	2.4	3.5	1.3	2.7	0.7	1.7	0.4	0.6
	7	154.0	1068.59	-	-	-	-	**-0.11**	-0.1	**-0.14**	-0.1
	8	135.0	1056.54	1.1	1.1	0.6	0.6	0.1	0.5	0.1	0.3
	1	1131.0	1031.07	1.8	2.0	1.2	1.5	1.2	1.6	0.9	1.2
	2	566.0	1030.45	1.5	1.6	1.2	1.3	1.6	1.7	0.4	0.9
	3	377.0	1031.59	1.1	1.5	0.6	0.9	1.1	1.2	0.9	1.1
	4	283.0	1031.07	1.0	1.3	0.5	0.8	0.6	1.1	0.5	0.9
	5	226.0	1030.86	1.0	1.5	0.2	0.5	1.0	1.4	0.3	0.7
	6	189.0	1030.45	0.9	1.6	0.8	1.3	1.1	1.3	0.4	0.8
	7	162.0	1036.08	1.1	1.6	0.2	0.6	**-0.33**	0.1	**-0.39**	0.1
	8	141.0	1044.32	1.3	1.5	0.7	0.8	0.2	0.6	0.1	0.3
CMT-5	1	1356.0	1302.43	0.8	1.1	0.6	0.8	0.5	1.1	0.1	0.3
	2	678.0	1302.15	0.8	1.2	0.5	0.9	0.8	1.1	0.3	0.6
	3	452.0	1301.29	1.0	1.3	0.6	1.0	0.5	1.2	0.1	0.7
	4	339.0	1304.78	0.7	1.2	0.3	0.6	1.0	1.2	0.2	0.4
	5	271.0	1300.02	1.4	1.8	1.3	1.5	1.0	1.3	0.5	1.1
	6	226.0	1303.37	0.8	1.0	0.7	0.9	1.1	1.4	0.0	0.6
	7	194.0	1309.40	0.1	1.2	0.1	0.4	**-0.32**	0.6	**-0.34**	0.3
	8	170.0	1303.91	1.1	1.7	0.7	1.4	**-0.06**	1.0	**-0.06**	0.7
	9	151.0	1307.93	2.0	2.8	1.2	1.7	0.3	1.3	0.2	1.2
	10	136.0	1323.01	2.2	2.2	1.0	1.0	0.0	0.5	**-0.57**	0.4
	1	1421.0	1299.86	2.8	3.0	2.7	2.8	2.7	3.0	1.8	2.4
	2	710.0	1305.35	2.1	2.5	2.1	2.2	2.3	2.4	1.6	1.9
	3	474.0	1301.03	2.0	2.4	2.0	2.3	2.2	2.6	1.3	1.8
	4	355.0	1303.65	1.4	1.7	1.3	1.6	2.4	2.7	1.6	1.7
	5	284.0	1300.62	1.6	2.0	1.3	1.8	2.4	2.7	1.4	1.7
	6	237.0	1306.17	1.7	1.9	1.5	1.6	1.9	2.2	0.8	1.2
	7	203.0	1301.54	2.0	2.3	1.4	1.7	2.5	2.6	1.3	1.6
	8	178.0	1308.78	1.2	1.4	0.8	1.2	0.6	1.3	0.1	0.8
	9	158.0	1307.25	1.3	1.7	0.7	1.2	0.6	1.1	**-0.23**	0.4
	10	142.0	1308.81	1.4	2.0	**-0.09**	1.0	0.6	0.7	**-0.29**	0.5

Table 9: Computational results with 1 000 000 iterations

The ALNSP algorithm produces high quality solutions for all benchmark instances known to be feasible. It also improves the best known solution values on 10 instances. Even if the ALNSM is outperformed by the ALNSP both in terms of feasibility and gaps, it still yields feasible solutions for 97 over the 99 instances in groups G1 and G2 and improves the best known solution values on two instances.

Table 10 reports the longest tour ratios (LTR) for the instances of group G3. The LTR of an R-MTVRP solution $\hat{\mathcal{S}}$ measures the degree of infeasibility of $\hat{\mathcal{S}}$ and is calculated by dividing the duration of the longest tour of $\hat{\mathcal{S}}$ by T_{max}. Table 10 shows the best value over five runs for both algorithm configurations and also reports the values provided by other authors when available. Both ALNS configurations produce values significantly smaller than those obtained by other authors, except for Olivera and Viera (2007).

6.4. Parameter analysis

We now provide some insights on specific design choices and values taken by the numerical parameters. Our aim is not to make an exhaustive analysis of all parameter values, but to focus on several aspects that have led to insightful observations during our experiments. In order to see how the different design choices and numerical parameter values affect the results, the configurations of both ALNS solution methods are performed while imposing constraints on parameters in `irace`, which leads to constrained configurations. For example, the set of available heuristics may be decided in advance by forcing some of the design choices to take *True* or *False* value. The parameters whose values are defined before the configuration through `irace` are said to be the "constrained parameters", while the others are

CVRP	m	TLG	BM	PS	SP	OV	ALNSM	ALNSP
CMT-1	3	1.115	1.041	1.026	1.030	**1.024**	1.028	1.029
CMT-1	4	**1.027**	**1.027**	1.085	1.056	**1.027**	**1.027**	1.033
CMT-2	7	1.073	1.088	1.060	1.102	1.009	1.008	**1.004**
CMT-12	6	1.064	1.072	1.029	1.029	**1.014**	1.017	**1.014**
F-11	3	1.075	**1.011**	1.020	1.027	1.020	1.020	1.020
Average		1.071	1.048	1.044	1.049	1.019	1.020	1.020

Table 10: G3: Longest tour ratios for various algorithms

called "free parameters". A set of constrained parameters with their respective values is called a "partial configuration". Since testing a high number of configurations for experimental purposes is time consuming, the analyses presented in this section were carried on the 52 benchmark instances with $T_{max} = T_1$ and with a modified stopping criterion (80 000 iterations and a time limit of $n^2/100$ seconds).

6.4.1. Average configurations

An "average configuration" is constructed for each ALNS algorithm, based on the corresponding seven final configurations (see Table 4). All design choices are set to the value that they take most often in the seven final configurations. All numerical parameters are set to the average of the seven values that they take in the final configurations (rounded if necessary). As mentioned in Section 6.2, the design choices nearly always take the same values over the seven final configurations for both ALNS heuristics. Since the design choices seem to be stable, testing the average configurations is interesting in order to see whether the value intervals matter more than the precise values in the case of numerical parameters.

We tested each final configuration and both average configurations using the modified stopping criterion. Five runs were performed on each of the 52 benchmark instances. For each configuration separately, the best and average gaps were recorded for each instance. In Table 11 we present the average of these gaps on the 52 benchmark instances as well as the number of unsolved instances for each configuration. The average number of runs yielding a feasible solution is reported in the second column. These results suggest that the performance of the algorithms is not very sensitive to variations of the numerical parameters as long as these are kept within a certain interval that can be evaluated by examining the final configurations for each algorithm. For a given solution method, the average configuration produces results similar to the ones of the final configurations.

Configuration	Unsolved	Average solved	Average best gap	Average gap
ALNSM-1	10	4.58	1.01	1.96
ALNSM-2	13	4.49	1.09	1.65
ALNSM-3	12	4.52	1.00	1.95
ALNSM-4	8	4.71	1.28	1.96
ALNSM-5	12	4.61	1.09	1.76
ALNSM-6	12	4.57	1.06	1.87
ALNSM-7	11	4.51	1.10	2.01
ALNSM-Avg	10	4.63	1.14	1.98
ALNSP-1	10	4.51	0.92	1.92
ALNSP-2	7	4.63	1.01	1.91
ALNSP-3	12	4.35	1.06	1.98
ALNSP-4	12	4.47	1.00	1.95
ALNSP-5	10	4.49	0.92	1.82
ALNSP-6	12	4.43	0.99	2.02
ALNSP-7	11	4.46	1.03	1.95
ALNSP-Avg	9	4.57	0.91	2.01

Table 11: Computational results for the final and average configurations of ALNSM and ALNSP

6.4.2. Heuristics

This section aims at understanding the behavior of the different heuristics introduced in Section 5.3. To this end, several sets of constrained configurations were produced:

- *Use of a single given removal heuristic.* For each removal heuristic in turn, the corresponding design choice is set to *True* and all the design choices related to other removal heuristics are set to *False*. This ensures that the constrained configurations produced by `irace` include only a single removal heuristic. The design choices related to insertion heuristics are free parameters.
- *Use of a single given insertion heuristic.* For each insertion heuristic in turn, the corresponding design choice is set to *True*, as well as that related to the greedy insertion. All design choices related to other insertion heuristics are set to *False*. The greedy insertion may be seen as a simplification of the regret heuristic, where only the best possible insertion position is considered; this is why it is set to *True* in all cases. The design choices related to removal heuristics are free parameters.

For each partial configuration passed to `irace`, we obtain seven constrained configurations. We have tested the average configurations obtained out of these constrained configurations. Table 12 shows the number of unsolved instances for each average constrained configuration.

	Removals									Insertions			
	Random removal	Worst removal (*a*)	Worst removal (*b*)	Worst removal (*c*)	Shaw removal	Historical removal (routes)	Historical removal (vehicles)	Cloud removal	Route removal	Greedy	Greedy + regret (routes)	Greedy + regret (vehicles)	Greedy + regret (positions)
ALNSM	16	13	23	10	9	18	17	26	44	17	12	21	12
ALNSP	16	14	-	16	10	18	-	21	-	14	11	-	12

Table 12: Number of unsolved instances for the average constrained configurations

As expected, the route removal heuristic performs very poorly when used on its own in the ALNSM method. In both algorithms, the Shaw removal and the worst removal heuristics perform significantly better than the other heuristics when used on their own, except for version *b* of the worst removal heuristic in the ALNSM algorithm which is outperformed by the random removal heuristic. Note that none of the other removal heuristics outperforms the random removal heuristic when used on its own.

For the ALNSP algorithm, all removal heuristics that performed better than the random removal heuristic when used separately were chosen by `irace` in the final configurations, and all others were discarded. For the ALNSM algorithm, `irace` discarded the *c* version of the worst removal heuristic and included the *b* version, but tests show that inversing these two choices produces similar results. Results deteriorate slightly in the case where only one version of the worst removal is included.

These observations suggest that ranking the heuristics based on their performances when used on their own should provide a good *a priori* indication of which ones are promising for the overall design of the ALNS and which ones are not. The same conclusion holds for the insertion heuristics. The regret heuristics based on positions or routes perform well when used on their own and both yield good results within the ALNSM algorithm, even if `irace` chooses only the route version in the final configurations. In fact, the quality of the results of the average ALNSM configuration improves slightly by adding the regret based on positions. The regret insertion heuristic based on vehicles yields poor results when used on its own with greedy. Again, for the ALNSP algorithm, `irace` chooses to keep only the regret heuristic based on routes, along with the greedy insertion heuristic.

6.4.3. Online and offline heuristic selection

The parameters of the roulette wheel mechanism adapt the heuristic selection process by allowing online changes in the heuristic selection probabilities. By extension, when all heuristics are available for selection by the roulette wheel, we say that the heuristic selection is performed online. This is the case when all the design choices corresponding to the inclusion of removal or insertion heuristics are set to *True* in the partial configuration passed to `irace`. In contrast, when these design choices are not constrained before the configuration phase, `irace` returns the set of heuristics available for the roulette wheel. In this case, we say that the heuristics are configured offline. The roulette wheel mechanism still plays a role in the online selection of the remaining heuristics but "unpromising" heuristics have already been discarded offline. Table 13 shows the sigma values for ALNSM and ALNSP configurations for offline and online heuristic selection. The offline configurations are the final ones. Note that since all numbers are rounded, the sum of the σ_ϕ values may not be equal to 1.

	ALNSM (offline)				ALNSM (online)				ALNSP (offline)				ALNSP (online)			
	σ_1	σ_2	$\sigma_1+\sigma_2$	σ_3	σ_1	σ_2	$\sigma_1+\sigma_2$	σ_3	σ_1	σ_2	$\sigma_1+\sigma_2$	σ_3	σ_1	σ_2	$\sigma_1+\sigma_2$	σ_3
1	0.48	0.40	0.88	0.12	0.29	0.71	1.00	0.00	0.19	0.28	0.46	0.54	0.26	0.70	0.96	0.04
2	0.29	0.42	0.71	0.29	0.44	0.51	0.95	0.05	0.06	0.16	0.22	0.78	0.30	0.56	0.86	0.14
3	0.42	0.27	0.68	0.32	0.16	0.81	0.97	0.03	0.07	0.22	0.29	0.71	0.06	0.42	0.47	0.53
4	0.17	0.03	0.20	0.80	0.13	0.71	0.84	0.16	0.05	0.01	0.06	0.94	0.12	0.77	0.90	0.10
5	0.40	0.33	0.72	0.28	0.27	0.61	0.88	0.12	0.06	0.33	0.38	0.62	0.32	0.67	0.99	0.01
6	0.30	0.39	0.68	0.32	0.10	0.82	0.91	0.09	0.16	0.59	0.75	0.25	0.20	0.28	0.48	0.52
7	0.16	0.25	0.41	0.59	0.22	0.75	0.97	0.03	0.27	0.46	0.73	0.27	0.11	0.74	0.85	0.15
Av.	0.31	0.30	0.61	0.39	0.23	0.70	0.93	0.07	0.12	0.29	0.41	0.59	0.20	0.59	0.79	0.21
Std.	0.12	0.13	0.23	0.23	0.13	0.12	0.06	0.06	0.08	0.19	0.26	0.26	0.10	0.18	0.22	0.22

Table 13: Sigma values for the offline and online ALNSM and ALNSP configurations

In both the ALNSM and ALNSP heuristics, the value of σ_3 is much smaller on average when all heuristics are available for selection by the roulette wheel (i.e., in the online configurations). These values suggest that intensification is preferred to diversification in the case where poor heuristic choices have not been discarded offline. This is especially true for the ALNSM. In fact, whether poor heuristics are discarded offline or not, the average value of σ_3 is significantly higher for the ALNSP. This is not surprising since a well-diversified population of routes must reinforce the AMR mechanism.

Once poor heuristic choices have been discarded offline, no general tendency emerges concerning the ordering of the σ_ϕ values, which may indicate that having three different σ_ϕ values does not have much importance. To test this hypothesis, we modified the average configurations of ALNSM and ALNSP to produce three alternative settings in which we fix the σ_ϕ values such that

- the intensification and diversification are equally rewarded: $\sigma_1 = \sigma_2 = \sigma_3$,
- the focus is put on intensification: $\sigma_1 = \sigma_2 = 0.4$ and $\sigma_3 = 0.2$,
- the focus is put on diversification: $\sigma_1 = \sigma_2 = 0.1$ and $\sigma_3 = 0.8$.

Moreover, for both ALNS heuristics, we also performed a test with complete removal of the roulette wheel selection mechanism: at each step, the removal and the insertion heuristics are chosen completely randomly. Table 14 summarizes the corresponding results.

	ALNSM		ALNSP	
	Unsolved	Average best gap	Unsolved	Average best gap
Average configuration, offline	10	0.95	9	0.98
One single sigma value, offline	13	1.02	12	0.91
Focus on intensification, offline	10	0.94	12	0.98
Focus on diversification, offline	12	1.13	9	1.08
No roulette wheel, offline	10	0.93	11	0.88
Average configuration, online	14	1.19	12	1.13
No roulette wheel, online	13	1.21	13	1.16

Table 14: Analysis of the roulette wheel mechanism

At first glance, it is unclear whether the focus should be set on intensification or on diversification when the heuristics selection has been carried out offline. Moreover, the complete removal of the roulette wheel mechanism does not seem to deteriorate the results. This suggests that with a careful offline heuristic selection, the roulette wheel mechanism does not need to be implemented to further guide the heuristic selection online. Note from Table 4 that both ρ and Θ, which respectively determine the persistence of roulette wheel knowledge and the roulette wheel time segment, take disparate values, tending to accredit the fact that the configuration of the roulette wheel mechanism has little influence on the numerical results of the algorithms.

The most surprising result is the following: when the selection of heuristics is performed online, the complete removal of the roulette wheel mechanism exhibits only a very slight deterioration of the results. However, the results obtained by the average configuration for the online case are clearly worse compared with those of the average configuration with an offline selection of heuristics by `irace`. Both in the case of ALNSM and ALNSP, doubling the parameters of the stopping criterion (i.e., doubling the number of iterations and the time limit) is necessary to obtain a similar quality of results, both in terms of feasibility and cost. This suggests that using the roulette wheel mechanism cannot effectively compensate for a poor offline selection of heuristics. We view this result as significant since it does not seem to have been previously reported in the ALNS literature. It is consistent with the extended experiments performed by Pellegrini et al. (2012) for the traveling salesman problem and the quadratic assignment problem solved with ant colony optimization algorithms. The authors show that adapting parameters online is ineffective as long as the value of these parameters is carefully determined offline.

6.4.4. ALNSP: the use of the routes memory

When looking at the design choices of the final configurations of the ALNSP, we see that the acceptance criterion is always the same: overtime is taken into account to determine the acceptance of new solutions (i.e., ALNSP(b) is chosen). Moreover, `irace` always chooses to set a relatively high number of iterations for the inner loop, implying a small number of calls to the AMR.

Table 15 shows the influence of $ALNS_{iter}$ on the results. For ALNSP(b), the average ALNSP configuration is modified by setting different $ALNS_{iter}$ values. In order to compare ALNSP(a) with ALNSP(b), a constrained configuration is passed to `irace`, imposing the ALNSP(a) acceptance criterion. An average configuration is deduced out of the seven ALNSP(a) configurations obtained. It is then modified by setting different $ALNS_{iter}$ values.

	ALNSP(b)		ALNSP(a)	
$ALNS_{iter}$	Unsolved	Average best gap	Unsolved	Average best gap
100	11	1.07	15	0.98
200	9	0.99	14	0.85
500	9	0.86	13	0.79
1000	9	0.72	14	0.74
10000	8	0.75	14	0.71
80000	8	0.82	15	0.79

Table 15: Influence of the number of ALNS iterations on the performance of ALNSP

In the case of ALNSP(b), the quality of the results seems relatively stable when the number of iterations of the inner loop varies as long as it is not too small. The best results are obtained with a few thousands iterations for the inner loop. This suggests that the memory is not the key element of the method, and it can certainly be seen more as an occasional perturbation. More experiments should be performed to assess whether other types of occasional perturbations may offer similar advantages with less computational effort. Note that the final configurations use the solution completion method where unrouted customers are placed into new routes. But in fact, our experiments show similar results when the second option is used for completion (see the end of Section 5.2.5), meaning that leaving the routes chosen from the memory unchanged when constructing a new initial solution for the inner loop of the ALNSP is not significantly important. This is consistent with the above observation, suggesting that the adaptive memory is not a key element for the success of the ALNSP.

For the ALNSP(a), the solution quality is good on average for feasible solutions but the number of unsolved instances clearly increases. We deduce that including overtime in the objective function to guide the search mechanism is much more crucial than the calls to the AMR. As is the case for ALNSP(b), the completion method where unrouted customers are placed into new routes is always chosen by `irace` for ALNSP(a), but this time, the results of the average configuration deteriorate if the alternative completion method is applied. This suggests that the overtime penalty mechanism included in the adaptive memory of routes is used to compensate the lack of overtime penalty in the objective function.

6.5. New best solutions

We provide in Table 16 new best solution values for 14 instances of G2. Note that some of the improved solutions of Table 7 are outperformed by those obtained with other configurations.

Instance	Previous best known	New best known	Configuration
CMT-2, $m = 6$, $T_{max} = 146$	858.58	855.34	ALNSP (250k), ALNSM (250k)
CMT-2, $m = 7$, $T_{max} = 131$	844.70	844.55	ALNSP, a final config. (250k)
CMT-3, $m = 5$, $T_{max} = 182$	832.34	831.20	ALNSP (250k)
CMT-4, $m = 5$, $T_{max} = 216$	1031.07	1029.65	ALNSM, a final config. (250k)
CMT-4, $m = 7$, $T_{max} = 154$	1068.59	1067.10	ALNSP (1000k)
CMT-4, $m = 7$, $T_{max} = 162$	1036.08	1032.07	ALNSP (1000k)
CMT-5, $m = 1$, $T_{max} = 1356$	1302.43	1298.35	ALNSP, a final config. (80k)
CMT-5, $m = 4$, $T_{max} = 339$	1304.78	1299.70	ALNSM, a final config. (250k)
CMT-5, $m = 7$, $T_{max} = 194$	1309.40	1304.02	ALNSP, a final config. (250k)
CMT-5, $m = 8$, $T_{max} = 170$	1303.91	1303.11	ALNSP (1000k)
CMT-5, $m = 10$, $T_{max} = 136$	1323.01	1315.47	ALNSP (1000k)
CMT-5, $m = 9$, $T_{max} = 158$	1307.25	1304.28	ALNSP (1000k)
CMT-5, $m = 10$, $T_{max} = 142$	1308.81	1305.01	ALNSP (1000k)
CMT-12, $m = 5$, $T_{max} = 172$	845.56	845.37	ALNSM, a final config. (80k)

Table 16: New best solution values

7. Conclusions

We have proposed multi-trip operators as an alternative to other approaches previously applied to the MTVRP. An extensive computational comparison was performed by developing and testing two ALNS algorithms. The first one, called ALNSM, includes multi-trip operators in heuristics that iteratively destroy and repair an R-MTVRP solution. The second one, called ALNSP, iteratively modifies a CVRP solution and applies bin packing techniques to assign the created routes to available vehicles, in order to produce R-MTVRP solutions. By testing numerically both methods on benchmark instances, we have shown that multi-trip operators deserve at the very least to be further investigated. Their integration into an ALNS framework yields high quality MTVRP solutions, close to the state-of-the-art results. Unlike the methods that combine VRP heuristics with BP techniques, multi-trip operators tend to slowly modify the route sequence of each vehicle. We believe that this may prove an advantage in more constrained problems, when the feasibility is highly dependent on customer sequences. The proposed ALNSP algorithm not only yields feasible solutions for all benchmark instances known to be feasible, but it also provides several new best solutions.

We used the automatic configuration tool `irace` to design and parametrize both solution methods, and experiments were performed to gain insight into the algorithm configurations. Separate evaluations of removal and insertion heuristics used in our ALNS algorithms suggest that the performance of a given heuristic, when applied on its own, is a good indicator of its effectiveness when included in the final design of the algorithm.

The study of the performance of the algorithms for several settings of the roulette wheel mechanisms suggests that it cannot on its own compensate for the absence of an offline heuristic selection. Moreover, when a careful offline selection of heuristics is performed, the contribution of the roulette wheel mechanism is seen not to be significant. The search mechanism of the ALNSP algorithm was also investigated and we have observed that integrating an R-MTVRP objective function (that penalizes overtime) is crucial, even when working on CVRP solutions. Moreover, the role of the adaptive memory does not seem to be critical for the success of the ALNSP heuristic.

Our experiments show the interest of using an automatic configuration tool, not only to improve the results obtained with a given algorithm, but also to gain knowledge into its components. We are convinced that systematically analyzing the behavior of several components is crucial because of the knowledge it generates, even though this is not sufficantly emphasized in the VRP literature.

Acknowledgments

This work has benefited from several discussions with the Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA), and in particular with Franco Mascia who granted support with `irace`. 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. Gilbert Laporte was funded by the Canadian National Sciences and Engineering Research Council under grant 2015-06189. Thanks are due to the referees for their valuable comments.

Appendix

Tables A1, A2, and A3 provide detailed computational results obtained with ALNSM and ALNSP on the 99 instances in G1 and G2 with 250 000 iterations. However this time, the best, worst, and average values over five runs are reported both in terms of total travel time and "z^*-gaps", as in Olivera and Viera (2007) and Cattaruzza et al. (2014b). For a given MTVRP solution, z^*gap denotes the percent deviation of the solution value compared to the value of z^* for the corresponding CVRP instance class. In Table A4, the solution values obtained with 1 000 000 iterations are given for instances of classes CMT-4 and CMT-5. Best known solution values are reported taking into account the new values given in Table 16.

References

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

Cattaruzza, D., Absi, N., Feillet, D., 2014a. The multi trip vehicle routing problem : a survey. Tech. Rep. EMSE CMPSFL 2014/3, CMP, Ecole des Mines de Saint-Étienne, Department of Manufacturing Sciences and Logistics.

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

Christofides, N., Mingozzi, A., Toth, P., 1979. The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, P. (Eds.), Combinatorial Optimization. Wiley, Chichester, pp. 315–338.

Diana, M., Dessouky, M. M., 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B: Methodological 38 (6), 539 – 557.

Fisher, M. L., 1994. Optimal solution of vehicle routing problems using minimum k-trees. Operations Research 42, 626–642.

Fleischmann, B., 1990. The vehicle routing problem with multiple use of vehicles. Tech. rep., Fachbereich Wirtschaftswissenschaften, Universität Hamburg.

Gendreau, M., Hertz, A., Laporte, G., 1992. New insertion and postoptimization procedures for the traveling salesman problem. Operations Research 40, 1086–1094.

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

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M., 2011. The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles.
URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf`

Mingozzi, A., Roberti, R., Toth, P., 2013. An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing 25 (2), 193 – 207.

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

Pellegrini, P., Birattari, M., Stützle., T., 2012. A critical analysis of parameter adaptation in ant colony optimization. Swarm Intelligence 6 (1), 23–48.

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

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

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66 (3), 331 – 340.

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

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

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

Salhi, S., Petch, R. J., 2007. A GA based heuristic for the vehicle routing problem with multiple trips. Journal of Mathematical Modelling and Algorithms 6, 591–613.

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. Vol. 1520 of Lecture Notes in Computer Science. pp. 417–431.

Taillard, E., 1993. Parallel iterative search methods for vehicle routing problems. Networks 23, 661–673.

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

Tillman, F., Cain, T., 1972. An upper bound algorithm for the single and multiple terminal delivery problem. Management Science 18, 664–682.

Appendix

Gr	CVRP	m	T_{max}	Best known	ALNSM (z^*-gaps)			ALNSP (z^*-gaps)			ALNSM (solution values)			ALNSP (solution values)		
					Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
G1	CMT-1	1	551	524.61	0.0	0.0	0.0	0.0	0.0	0.0	524.61	524.61	524.61	524.61	524.61	524.61
G1	$z^* = 524.61$	2	275	533.00	2.2	4.4	3.0	1.6	3.5	2.6	536.37	547.87	540.56	533.00	543.05	538.15
G1		1	577	524.61	0.0	0.0	0.0	0.0	0.0	0.0	524.61	524.61	524.61	524.61	524.61	524.61
G1		2	289	529.85	1.0	1.0	1.0	1.0	1.0	1.0	529.85	529.85	529.85	529.85	529.85	529.85
G2		3	192	552.68	5.9	6.6	6.1	5.9	6.4	6.1	555.57	559.19	556.49	555.57	558.08	556.58
G1		4	144	546.29	6.5	6.5	6.5	4.1	6.0	5.3	558.64	558.64	558.64	546.29	556.21	552.23
G1	CMT-2	1	877	835.26	0.3	0.7	0.5	0.0	0.5	0.2	837.52	841.13	839.16	835.41	839.77	837.07
G1	$z^* = 835.26$	2	439	835.26	0.2	0.7	0.4	0.0	0.1	0.0	836.71	841.13	839.01	835.26	835.77	835.46
G1		3	292	835.26	0.1	0.6	0.3	0.0	0.6	0.1	835.89	839.87	837.95	835.26	840.15	836.37
G1		4	219	835.26	0.0	0.6	0.3	0.0	0.4	0.1	835.26	840.56	837.35	835.28	838.60	835.95
G1		5	175	835.80	1.0	2.5	1.7	0.1	0.2	0.1	843.70	856.29	849.71	836.18	836.81	836.41
G2		6	146	855.34	2.4	2.4	2.4	2.4	2.8	2.5	855.34	855.34	855.34	855.34	858.58	856.15
G1		1	919	835.26	0.1	0.9	0.5	0.0	0.2	0.1	835.89	842.80	839.48	835.26	837.24	836.43
G1		2	459	835.26	0.0	0.6	0.3	0.1	0.2	0.1	835.26	840.14	838.05	835.77	836.78	836.17
G1		3	306	835.26	0.2	0.7	0.5	0.0	0.2	0.1	836.62	840.99	839.44	835.28	836.81	836.01
G1		4	230	835.26	0.0	0.8	0.4	0.0	0.1	0.1	835.28	842.21	838.62	835.26	835.89	835.74
G1		5	184	835.26	0.0	0.5	0.2	0.0	0.1	0.0	835.26	839.35	836.91	835.26	835.89	835.65
G1		6	153	839.22	0.6	1.7	1.2	0.5	0.5	0.5	840.65	849.28	844.89	839.22	839.22	839.22
G2		7	131	844.55	2.4	3.2	2.9	1.5	1.8	1.7	855.63	861.70	859.42	848.20	850.70	849.27
G1	CMT-3	1	867	826.14	0.2	0.5	0.4	0.0	0.3	0.1	827.85	830.64	829.64	826.14	828.67	827.35
G1	$z^* = 826.14$	2	434	826.14	0.4	0.5	0.5	0.2	0.4	0.3	829.37	830.40	829.95	827.53	829.63	828.90
G1		3	289	826.14	0.4	0.6	0.5	0.0	0.4	0.3	829.75	830.94	830.36	826.14	829.45	828.84
G2		4	217	829.45	0.7	1.7	1.1	0.4	0.5	0.4	831.60	839.93	835.44	829.45	830.03	829.62
G2		5	173	832.89	2.5	3.1	2.8	0.8	2.4	1.3	847.19	852.08	849.62	832.89	845.63	837.10
G2		6	145	836.22	2.7	4.1	3.4	1.2	4.3	2.7	848.52	859.72	853.99	836.22	862.07	848.60
G1		1	909	826.14	0.5	0.9	0.7	0.2	0.4	0.3	830.27	833.68	832.31	827.39	829.63	828.66
G1		2	454	826.14	0.4	0.6	0.5	0.0	0.4	0.3	829.66	831.04	830.30	826.14	829.63	828.93
G1		3	303	826.14	0.3	0.6	0.5	0.0	0.4	0.1	828.87	831.40	830.29	826.14	829.45	827.22
G1		4	227	826.14	0.5	0.9	0.7	0.0	0.4	0.3	830.11	833.56	832.14	826.14	829.63	828.87
G2		5	182	831.20	0.8	1.0	0.9	0.6	0.9	0.8	832.89	834.20	833.25	831.20	833.79	832.46
G2		6	151	834.35	1.2	1.3	1.2	1.0	1.2	1.1	836.07	836.95	836.40	834.53	836.30	835.27

Table A1: z^*-gaps and solution values for 250k iterations for instance classes CMT-1, CMT-2 and CMT-3

Gr	CVRP	m	T_{max}	Best known	ALNSM (z^*-gaps)			ALNSP (z^*-gaps)			ALNSM (solution values)			ALNSP (solution values)		
					Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
G2	CMT-4	1	1080	1031.00	1.1	1.3	1.2	0.8	1.4	1.2	1039.62	1042.11	1040.57	1036.80	1042.48	1040.38
G2	$z^* = 1028.42$	2	540	1031.07	0.6	1.5	1.0	1.0	1.2	1.1	1035.00	1043.75	1038.89	1039.04	1041.15	1040.18
G2		3	360	1028.42	0.7	1.2	1.0	0.2	1.2	0.8	1035.11	1041.12	1038.59	1029.99	1040.93	1036.68
G2		4	270	1031.10	1.1	1.6	1.3	0.9	1.7	1.3	1040.11	1045.38	1042.26	1037.29	1045.78	1041.73
G2		5	216	1029.65	1.6	3.5	2.4	0.9	1.5	1.1	1044.91	1064.69	1053.47	1037.71	1043.78	1039.82
G2		6	180	1034.61	3.0	4.7	4.1	1.3	2.8	2.3	1058.94	1076.44	1070.81	1041.57	1057.23	1051.85
G2		7	154	1067.10	-	-	-	3.8	3.8	3.8	-	-	-	1067.10	1067.10	1067.10
G2		8	135	1056.54	3.9	3.9	3.9	2.9	3.6	3.2	1068.09	1068.09	1068.09	1058.52	1065.40	1061.41
G2		1	1131	1031.07	2.1	2.4	2.3	1.5	2.1	1.9	1049.98	1053.09	1052.12	1043.85	1050.18	1047.74
G2		2	566	1030.45	1.7	2.0	1.8	1.8	2.0	1.9	1046.02	1048.90	1047.06	1047.03	1048.96	1048.30
G2		3	377	1031.59	1.4	2.1	1.8	1.4	1.7	1.6	1043.24	1049.52	1047.25	1042.43	1045.72	1044.37
G2		4	283	1031.07	1.3	1.7	1.6	0.8	1.6	1.4	1041.72	1046.17	1044.53	1036.82	1045.23	1042.76
G2		5	226	1030.86	1.2	2.0	1.8	1.3	1.8	1.7	1041.12	1049.28	1046.59	1041.32	1047.40	1045.56
G2		6	189	1030.45	1.1	2.1	1.8	1.3	1.8	1.5	1039.37	1049.89	1046.92	1041.81	1046.68	1043.63
G2		7	162	1032.07	1.8	3.3	2.3	0.4	1.3	0.9	1046.98	1061.89	1052.48	1032.87	1041.94	1037.56
G2		8	141	1044.32	2.8	3.5	3.1	1.7	2.4	2.1	1057.63	1064.44	1060.04	1046.25	1052.67	1050.38
G2	CMT-5	1	1356	1298.35	1.7	2.3	1.9	1.4	2.2	1.9	1313.50	1320.69	1316.51	1309.00	1320.33	1316.15
G2	$z^* = 1291.44$	2	678	1302.15	1.6	2.3	2.1	1.6	2.2	1.9	1312.29	1320.72	1318.19	1312.34	1319.88	1316.07
G2		3	452	1301.29	1.8	2.2	2.0	1.3	2.4	2.0	1314.09	1320.22	1317.66	1307.84	1321.88	1317.35
G2		4	339	1299.70	1.7	2.4	2.3	2.1	2.7	2.2	1313.92	1322.99	1320.56	1318.18	1326.24	1320.20
G2		5	271	1300.02	2.1	3.0	2.5	1.6	2.3	2.0	1318.23	1330.18	1323.37	1312.70	1321.26	1317.52
G2		6	226	1303.37	1.8	2.3	1.9	2.0	2.7	2.3	1314.34	1320.89	1315.90	1317.33	1326.70	1321.55
G2		7	194	1304.02	1.5	3.1	2.7	1.0	2.4	2.0	1311.14	1332.11	1325.67	1304.83	1322.95	1317.57
G2		8	170	1303.11	2.1	3.2	2.7	0.9	2.7	2.0	1318.30	1332.55	1326.43	1303.11	1325.85	1317.20
G2		9	151	1307.93	3.3	4.6	4.1	1.5	4.0	2.6	1333.44	1350.38	1344.76	1311.45	1342.82	1325.44
G2		10	136	1323.01	4.7	4.7	4.7	2.5	3.4	2.9	1352.43	1352.43	1352.43	1323.30	1335.03	1329.40
G2		1	1421	1299.86	3.5	3.9	3.7	3.4	3.9	3.7	1336.33	1342.05	1339.43	1335.58	1341.29	1339.40
G2		2	710	1305.35	3.2	3.8	3.6	3.4	3.7	3.5	1332.97	1341.06	1337.75	1335.30	1339.68	1337.24
G2		3	474	1301.03	2.7	3.5	3.2	2.9	3.9	3.4	1326.57	1336.21	1332.52	1329.52	1341.97	1335.34
G2		4	355	1303.65	2.4	2.9	2.6	3.4	3.9	3.7	1322.37	1329.28	1325.21	1334.85	1342.15	1338.68
G2		5	284	1300.62	2.3	3.0	2.7	3.2	3.8	3.4	1321.10	1330.50	1326.57	1332.20	1340.63	1335.92
G2		6	237	1306.17	2.8	3.4	3.1	3.1	3.7	3.4	1327.93	1334.81	1331.43	1331.37	1338.90	1334.79
G2		7	203	1301.54	2.8	3.3	3.1	3.3	3.5	3.4	1328.08	1334.38	1332.03	1333.84	1336.02	1334.73
G2		8	178	1308.78	2.5	3.1	2.8	1.9	3.0	2.7	1324.14	1331.26	1327.35	1315.98	1330.71	1325.78
G2		9	158	1304.28	2.6	3.1	2.9	1.8	2.7	2.3	1324.40	1331.82	1329.46	1314.48	1326.46	1321.48
G2		10	142	1305.01	2.8	3.7	3.4	1.9	2.4	2.1	1327.28	1339.72	1334.89	1316.31	1322.24	1318.62

Table A2: z^*-gaps and solution values for 250k iterations for instance classes CMT-4 and CMT-5

Gr	CVRP	m	T_{max}	Best known	ALNSM (z^*-gaps)			ALNSP (z^*-gaps)			ALNSM (solution values)			ALNSP (solution values)		
					Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
G1	CMT-11	1	1094	1042.11	0.0	0.0	0.0	0.0	0.4	0.2	1042.12	1042.42	1042.36	1042.12	1046.50	1043.91
G1	$z^* = 1042.11$	2	547	1042.11	0.4	0.6	0.4	0.1	2.9	0.9	1045.93	1047.94	1046.64	1043.11	1072.06	1051.67
G1		3	365	1042.11	0.0	0.0	0.0	0.0	0.7	0.2	1042.12	1042.42	1042.24	1042.12	1049.03	1044.66
G2		4	274	1078.64	3.5	4.7	4.0	3.5	4.2	3.9	1078.64	1091.03	1083.38	1078.74	1085.75	1082.24
G1		5	219	1042.11	0.0	0.2	0.1	0.0	0.3	0.1	1042.12	1043.89	1042.80	1042.12	1044.76	1043.04
G1		1	1146	1042.11	0.0	0.1	0.1	0.0	2.9	1.8	1042.12	1043.18	1042.74	1042.12	1072.51	1061.11
G1		2	573	1042.11	0.0	0.1	0.1	0.0	0.7	0.3	1042.12	1043.55	1042.81	1042.12	1049.18	1045.13
G1		3	382	1042.11	0.0	0.0	0.0	0.0	0.0	0.0	1042.12	1042.42	1042.18	1042.12	1042.12	1042.12
G1		4	287	1042.11	0.0	0.2	0.1	0.0	0.0	0.0	1042.12	1043.86	1043.03	1042.12	1042.12	1042.12
G1		5	229	1042.11	0.0	0.2	0.1	0.0	0.0	0.0	1042.12	1043.85	1042.91	1042.12	1042.12	1042.12
G1	CMT-12	1	861	819.56	0.0	0.0	0.0	0.0	0.0	0.0	819.56	819.56	819.56	819.56	819.56	819.56
G1	$z^* = 819.56$	2	430	819.56	0.6	0.6	0.6	0.0	0.0	0.0	824.78	824.78	824.78	819.56	819.56	819.56
G1		3	287	819.56	0.6	2.7	2.1	0.0	0.0	0.0	824.78	841.38	837.02	819.56	819.56	819.56
G1		4	215	819.56	0.0	0.1	0.0	0.0	0.0	0.0	819.56	820.48	819.87	819.56	819.56	819.56
G2		5	172	845.37	-	-	-	3.2	3.2	3.2	-	-	-	845.56	845.56	845.56
G1		1	902	819.56	0.0	0.0	0.0	0.0	0.0	0.0	819.56	819.56	819.56	819.56	819.56	819.56
G1		2	451	819.56	0.0	0.0	0.0	0.0	0.0	0.0	819.56	819.56	819.56	819.56	819.56	819.56
G1		3	301	819.56	0.0	0.0	0.0	0.0	0.0	0.0	819.56	819.56	819.56	819.56	819.56	819.56
G1		4	225	819.56	0.0	0.6	0.1	0.0	0.0	0.0	819.56	824.78	820.60	819.56	819.56	819.56
G1		5	180	824.78	0.6	0.9	0.7	0.6	0.6	0.6	824.78	826.86	825.61	824.78	824.78	824.78
G1		6	150	823.14	2.8	6.6	4.9	0.6	4.3	3.6	842.90	873.46	859.52	824.16	854.82	848.69
G2	F-11	1	254	241.97	0.0	0.0	0.0	0.0	3.2	1.3	241.97	241.97	241.97	241.97	249.66	245.05
G2	$z^* = 241.97$	2	127	250.85	3.7	3.7	3.7	3.7	3.7	3.7	250.85	250.85	250.85	250.85	250.85	250.85
G2		1	266	241.97	0.0	0.0	0.0	0.0	0.0	0.0	241.97	241.97	241.97	241.97	241.97	241.97
G2		2	133	241.97	4.8	4.8	4.8	0.0	0.0	0.0	253.56	253.56	253.56	241.97	241.97	241.97
G2		3	89	254.07	5.3	6.2	5.6	5.0	5.0	5.0	254.90	256.87	255.64	254.07	254.07	254.07
G2	F-12	1	1221	1162.96	0.9	1.2	1.0	0.1	2.1	1.2	1173.65	1176.51	1174.70	1164.51	1187.57	1176.71
G2	$z^* = 1162.96$	2	611	1162.96	0.3	0.7	0.4	0.1	0.9	0.4	1166.25	1170.54	1167.39	1164.25	1173.89	1167.97
G2		3	407	1162.96	2.6	4.4	3.5	0.1	2.5	1.2	1193.23	1214.29	1203.11	1164.25	1191.58	1176.69
G2		1	1279	1162.96	0.4	0.7	0.5	0.0	1.0	0.5	1167.23	1171.52	1168.93	1163.30	1174.93	1169.02
G2		2	640	1162.96	0.0	0.4	0.3	0.3	1.2	0.9	1163.30	1167.09	1165.94	1166.52	1176.62	1172.91
G2		3	426	1162.96	2.1	2.1	2.1	0.1	1.1	0.8	1187.57	1187.86	1187.74	1164.25	1175.19	1171.95

Table A3: z^*-gaps and solution values for 250k iterations for instance classes CMT-11, CMT-12, F-11 and F-12

Gr	CVRP	m	T_{max}	Best known	ALNSM (z^*-gaps)			ALNSP (z^*-gaps)			ALNSM (solution values)			ALNSP (solution values)		
					Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
G2	CMT-4	1	1080	1031.00	0.9	1.2	1.0	0.5	1.2	0.9	1039.62	1042.11	1040.57	1036.80	1042.48	1040.38
G2	$z^* = 1028.42$	2	540	1031.07	0.5	0.6	0.5	0.3	0.9	0.6	1035.00	1043.75	1038.89	1039.04	1041.15	1040.18
G2		3	360	1028.42	0.4	0.7	0.6	0.1	0.7	0.4	1035.11	1041.12	1038.59	1029.99	1040.93	1036.68
G2		4	270	1031.10	0.3	1.2	0.9	0.2	0.9	0.7	1040.11	1045.38	1042.26	1037.29	1045.78	1041.73
G2		5	216	1029.65	0.4	2.8	1.5	0.2	0.8	0.6	1044.91	1064.69	1053.47	1037.71	1043.78	1039.82
G2		6	180	1034.61	1.9	4.2	3.3	1.0	1.4	1.2	1058.94	1076.44	1070.81	1041.57	1057.23	1051.85
G2		7	154	1067.10	-	-	-	3.8	3.8	3.8	-	-	-	1067.10	1067.10	1067.10
G2		8	135	1056.54	3.3	3.3	3.3	2.9	3.2	3.1	1068.09	1068.09	1068.09	1058.52	1065.40	1061.41
G2		1	1131	1031.07	1.4	2.0	1.8	1.2	1.7	1.5	1049.98	1053.09	1052.12	1043.85	1050.18	1047.74
G2		2	566	1030.45	1.4	1.6	1.5	0.6	1.4	1.1	1046.02	1048.90	1047.06	1047.03	1048.96	1048.30
G2		3	377	1031.59	0.9	1.5	1.2	1.2	1.5	1.4	1043.24	1049.52	1047.25	1042.43	1045.72	1044.37
G2		4	283	1031.07	0.8	1.4	1.1	0.7	1.5	1.1	1041.72	1046.17	1044.53	1036.82	1045.23	1042.76
G2		5	226	1030.86	0.5	1.1	0.7	0.5	1.3	1.0	1041.12	1049.28	1046.59	1041.32	1047.40	1045.56
G2		6	189	1030.45	1.0	1.7	1.5	0.6	1.2	1.0	1039.37	1049.89	1046.92	1041.81	1046.68	1043.63
G2		7	162	1032.07	1.0	1.9	1.4	0.4	1.2	0.8	1046.98	1061.89	1052.48	1032.87	1041.94	1037.56
G2		8	141	1044.32	2.2	2.5	2.3	1.7	2.3	1.8	1057.63	1064.44	1060.04	1046.25	1052.67	1050.38
G2	CMT-5	1	1356	1298.35	1.4	1.9	1.6	0.9	1.6	1.2	1313.50	1320.69	1316.51	1309.00	1320.33	1316.15
G2	$z^* = 1291.44$	2	678	1302.15	1.3	2.0	1.8	1.2	1.7	1.4	1312.29	1320.72	1318.19	1312.34	1319.88	1316.07
G2		3	452	1301.29	1.4	1.9	1.7	0.8	1.9	1.4	1314.09	1320.22	1317.66	1307.84	1321.88	1317.35
G2		4	339	1299.70	1.3	1.9	1.7	1.2	1.5	1.4	1313.92	1322.99	1320.56	1318.18	1326.24	1320.20
G2		5	271	1300.02	2.0	2.5	2.2	1.1	2.3	1.8	1318.23	1330.18	1323.37	1312.70	1321.26	1317.52
G2		6	226	1303.37	1.6	2.0	1.9	0.9	1.9	1.5	1314.34	1320.89	1315.90	1317.33	1326.70	1321.55
G2		7	194	1304.02	1.5	1.8	1.7	1.0	2.1	1.7	1311.14	1332.11	1325.67	1304.83	1322.95	1317.57
G2		8	170	1303.11	1.7	3.0	2.4	0.9	2.5	1.6	1318.30	1332.55	1326.43	1303.11	1325.85	1317.20
G2		9	151	1307.93	2.5	3.5	3.0	1.5	3.6	2.5	1333.44	1350.38	1344.76	1311.45	1342.82	1325.44
G2		10	136	1323.01	3.5	3.5	3.5	1.9	3.2	2.9	1352.43	1352.43	1352.43	1323.30	1335.03	1329.40
G2		1	1421	1299.86	3.3	3.7	3.5	2.5	3.3	3.0	1336.33	1342.05	1339.43	1335.58	1341.29	1339.40
G2		2	710	1305.35	3.2	3.4	3.3	2.7	3.4	3.0	1332.97	1341.06	1337.75	1335.30	1339.68	1337.24
G2		3	474	1301.03	2.8	3.2	3.0	2.1	3.0	2.6	1326.57	1336.21	1332.52	1329.52	1341.97	1335.34
G2		4	355	1303.65	2.2	2.9	2.6	2.5	2.9	2.7	1322.37	1329.28	1325.21	1334.85	1342.15	1338.68
G2		5	284	1300.62	2.0	2.8	2.5	2.1	2.7	2.4	1321.10	1330.50	1326.57	1332.20	1340.63	1335.92
G2		6	237	1306.17	2.6	2.9	2.7	1.9	2.6	2.4	1327.93	1334.81	1331.43	1331.37	1338.90	1334.79
G2		7	203	1301.54	2.2	2.8	2.5	2.0	2.8	2.4	1328.08	1334.38	1332.03	1333.84	1336.02	1334.73
G2		8	178	1308.78	2.1	2.7	2.5	1.5	2.6	2.1	1324.14	1331.26	1327.35	1315.98	1330.71	1325.78
G2		9	158	1304.28	2.0	2.8	2.4	1.0	2.1	1.6	1324.40	1331.82	1329.46	1314.48	1326.46	1321.48
G2		10	142	1305.01	1.3	3.0	2.4	1.1	2.0	1.8	1327.28	1339.72	1334.89	1316.31	1322.24	1318.62

Table A4: z^*-gaps and solution values for 1000k iterations for instance classes CMT-4 and CMT-5