Improving neighborhood exploration into MOEA/D framework to solve a bi-objective routing problem

Clément Legrand, Diego Cattaruzza, Laetitia Jourdan, Marie-Eléonore Kessaci

Improving Neighborhood Exploration into MOEA/D Framework to Solve a Bi-Objective Routing Problem

Clément Legrand[a], Diego Cattaruzza[b], Laetitia Jourdan[a], Marie-Eléonore Kessaci[a]

[a]*Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
[b]*Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
E-mail: clement.legrand4.etu@univ-lille.fr [C. Legrand]; diego.cattaruzza@centralelille.fr [D. Cattaruzza]; laetitia.jourdan@univ-lille.fr [L. Jourdan];marie-eleonore.kessaci@univ-lille.fr [M-E. Kessaci]

Abstract

Local search (LS) algorithms are efficient metaheuristics to solve combinatorial problems. The performance of LS highly depends on the neighborhood exploration of solutions. Many methods have been developed over the years to improve the efficiency of LS on different problems of operations research. In particular, the exploration strategy of the neighborhood and the exclusion of irrelevant neighboring solutions are design mechanisms that have to be carefully considered when tackling NP-hard optimization problems. A MOEA/D framework including an LS-based mutation and knowledge discovery mechanisms is the core algorithm used to solve a bi-objective vehicle routing problem with time windows (bVRPTW) where the total traveling cost and the total waiting time of drivers have to be minimized. We enhance the classical LS exploration strategy of the neighborhood from the literature of scheduling and propose new metrics based on customers distances and waiting times to reduce the neighborhood size. We conduct a deep analysis of the parameters to give a fine-tuning of the MOEA/D framework adapted to the LS variants and to the bVRPTW. Experiments show that the proposed neighborhood strategies lead to better performance on both Solomon's and Gehring and Homberger's benchmarks.

1. Introduction

With the increasing demand in terms of services (transportation, delivery, storage), comes an explosion of new challenges that have to be faced. Most of them require taking into account several conflicting aspects (ecological, economical, societal), each one characterized by an objective function to optimize. When at least two objectives are optimized simultaneously, we are facing a multi-criterion decision. In some cases, the decision maker provides a score of interest for each objective, which can lead to a reduction of the multi-objective problem to a single-objective problem where the different objectives are aggregated. Unfortunately, most of the time, the decision maker does not know precisely how the objectives interact with each other, which may lead to a bad strategy to solve the problem, and consequently to irrelevant results.

In the context of this paper, we are interested in the Vehicle Routing Problem with Time Windows (VRPTW). It can be considered a logistic problem, where vehicles have to serve customers within a precise time interval. This problem has been introduced decades ago, and many objectives have been investigated.

However, to the best of our knowledge, only a small subset of these objectives has received enough interest to have publicly available results. Thus, it is very hard to know if an algorithm or a method is competitive on the problem itself (independent of the objectives chosen) or if it is only relevant to the objectives studied. To overcome this lack in the literature, we propose to focus on a bi-objective VRPTW (bVRPTW) where one objective has been highly studied, that is the total transportation cost, and an objective that has rarely been studied, that is, the total waiting time incurred when drivers arrive before the opening of the time window (Castro-Gutierrez et al., 2011; Zhou and Wang, 2014). Using both of these objectives is interesting in many real-life situations, like food delivery, where the waiting time of a driver impacts the heat of the meals of the next customers, and consequently customer satisfaction. Another typical situation concerns the transportation of people, more precisely when a patient has a medical appointment, we do not want that he/she waits too much.

To solve the problem, we use MOEA/D, a Multi-Objective Evolutionary Algorithm based on Decomposition (Zhang and Li, 2007) where the mutation step is replaced by a local search, and enhanced with Knowledge Discovery (KD) (Legrand et al., 2022). Local searches (LS) are known to be powerful algorithms used in evolutionary algorithms to improve their performance (Knowles, 2002). Indeed, LS are able to intensify the search by focusing on a specific region of the search space. They are based on neighborhood operators that link solutions together and a neighborhood exploration strategy defines how the neighbors are explored and when the exploration is stopped. Many LS have been developed, and many operators are available, for this problem, but most of them only consider a single objective which is the total transportation cost. Moreover, LS steps are quite time-consuming, which is why different strategies exist to speed up the search and reduce the time allocated to neighborhood exploration. Indeed, we can explore more or less the neighborhood according to the strategy used. With a *first* strategy, the first improving solution met is kept, allowing a fast exploration of the space but with a relatively long path to a local optimum. While a *best* strategy explores the whole neighborhood to apply the best improving move, allowing a smaller path but the exploration itself is more time-consuming. Since routing problems produce large neighborhoods, reduction techniques have been designed to avoid irrelevant moves. The most common one is probably the granular search introduced by Toth and Vigo (2003). It is based on the idea that two *distant* customers have a low chance to be served one right after the other in the same route. However, it requires a proper definition of the distance between customers, which may not be obvious when several objectives are considered and are not in the same range of values.

The contribution of the paper is to enhance the basic local search used in MOEA/D considering new neighborhood mechanisms. First, we present a new strategy to explore the neighborhood of bVRPTW solutions inspired by the state of the art for permutation flow-shop scheduling problem Ruiz and Stützle (2007). The neighborhood is then divided into subsets of neighbors. Contrary to the *best* strategy that explores the neighborhood entirely before making a decision or the *first* strategy that makes a decision for each visited neighbor, the proposed strategy, called *first-best*, explores a subset entirely before making a decision. Second, we define a new metric that considers not only the Euclidean distance between the customers but also their respective time windows to evaluate the waiting time. With such a metric, we hope to obtain a more accurate neighborhood, depending on the two objectives optimized. Two variants of the metric are proposed. One variant aggregates the cost and the waiting time with variable weights, while the other uses the same weights for the aggregation. These contributions are supported by a rigorous analysis of the different parameters used in the MOEA/D framework, through irace (López-Ibáñez et al., 2016), to highlight the most influential ones. Therefore, each variant tested is then tuned with irace to keep the configuration the best adapted for each one. Experiments are conducted on Solomon's benchmark with instances of size 100, and on Gehring and Homberger's benchmark with instances of size 200 to evaluate the performance of each MOEA/D variant and define the best-suited neighborhood mechanisms.

The paper is structured as follows. Section 2 introduces the problem and motivates our choice of ob-

jectives to optimize. Concepts related to multi-objective optimization are presented in Section 3. Our contribution concerning the local search, as well as the algorithms compared, are detailed in Section 4. The experimental setup (i.e., benchmarks, termination criterion, performance assessment, and machines used) is described in Section 5. Our experimental protocol, concerning the tuning and the evaluation of the performances, is presented in Section 6. The tuning is analyzed in Section 7, and the results obtained are discussed in Section 8. We conclude the paper in Section 9.

2. Description of the Problem

2.1. Vehicle Routing Problem with Time Windows (VRPTW)

The VRPTW (Toth and Vigo, 2014) is defined on a graph $G = (V, E)$, where $V = \{0, 1, \ldots, N\}$ is the set of vertices and $E = \{(i, j) \mid i, j \in V\}$ is the set of arcs. It is possible to travel from i to j, incurring a travel cost c_{ij} and a travel time t_{ij}. Usually, the cost is computed as the Euclidean distance between the customers. Vertex 0 represents the depot where a fleet of K identical vehicles with limited capacity $\mathcal{Q}$ is based. Vertices $1, \ldots, N$ represent the customers to be served, each one having a demand q_i, a time window $[a_i, b_i]$ during which service must occur, and a service time s_i estimating the required time to perform the delivery. Vehicles may arrive before a_i. In that case, the driver has to wait until a_i to accomplish service incurring a waiting time. Arriving later than b_i is not allowed (the time windows are *hard*). We recall that a *route* r is an elementary cycle on G that contains the depot (that is vertex 0) and is expressed as a sequence of vertices $r = (v_0, v_1, \ldots, v_{|r|}, v_{|r|+1})$ where $v_0 = v_{|r|+1} = 0$ and vertices $v_1, \ldots, v_{|r|}$ are all different. The cost c_r of a route r is then given as the sum of traveling costs on arcs used to visit subsequent vertices, that is $\sum_{i=0}^{|r|} c_{v_i, v_{i+1}}$. A solution x is represented as a set of K (possibly empty) routes, that is $x = \{r_1, \ldots, r_K\}$, and its cost is expressed as:

$$f_1(x) = \sum_{k=1}^{K} c_{r_k} \tag{1}$$

The VRPTW calls for the determination of at most K routes such that the traveling cost is minimized and the following conditions are satisfied: (a) each route starts and ends at the depot, (b) each customer is visited by exactly one route, (c) the sum of the demands of the customers in any route does not exceed $\mathcal{Q}$, (d) time windows are respected.

2.2. Objectives Related to the VRPTW

A wide range of objectives has already been investigated in the VRPTW context. This section provides a broad overview of the most common objectives found in the literature. Following the classification of Jozefowiez et al. (2008), these objectives are divided into three categories according to the component of the problem they are associated with the route, the node or arc activity, and the resources.

2.2.1. Objectives Associated to the Route

Among the objectives related to the route, the most common is surely minimizing the cost of the solutions generated (Schneider et al., 2017). For instance, the cost can be the distance traveled or the time required. It can also be something more complex, like CO_2 emissions. More generally, minimizing cost is linked to an economic criterion. When the total cost is ignored, the makespan (i.e., minimizing the length of the longest

route) is an objective frequently minimized (Zhou and Wang, 2014). These objectives are motivated by the applications of the problem. For example, if electric vehicles are used, the length of the tours should not exceed the autonomy of the vehicles.

2.2.2. Objectives Associated to Nodes or Arcs

Concerning the objectives related to the node or arc activity, they often involve time windows, since it is the most constraining part of the problem in general. When considering *hard* time windows (i.e., that cannot be violated), the driver's waiting time due to earliness can be optimized (Zhang et al., 2019). If we consider *soft* time windows (i.e., that can be violated with a penalty), the delay time of drivers is more likely to be optimized (Zhou and Wang, 2014). More generally, the number of violated time windows can also be minimized (Geiger, 2003).

2.2.3. Objectives Associated to Resources

The last category of objectives concerns resources. The main resources encountered in the literature are vehicles and goods. The minimization of the number of vehicles often appears in classical benchmarks and can be interpreted economically, in that fewer vehicles means less monetary investment. More precisely, this objective is often minimized first in Solomon's and Gehring and Homberger's benchmarks (Solomon, 1987; Gehring and Homberger, 1999), since buying a truck or hiring a driver is the most expensive part. Finally, some objectives try to erase disparities between tours, to bring a *fairness* aspect into the problem. To define a balancing objective it is necessary to define a route's workload, which can be expressed as the number of customers visited or the number of goods delivered (Melián-Batista et al., 2014; Baños et al., 2013).

2.3. Minimizing the Waiting Time of Drivers

When drivers arrive before the opening of a time window they must wait until the opening time. It increases the time of the route for the driver and may incur satisfaction issues. The total waiting time of drivers is an objective that has not been highly studied in the literature. In the last decade, Castro-Gutierrez et al. (2011) studied how this objective was correlated to four other common objectives in the VRPTW context. In particular, they showed that in Solomon's instances, the total traveled distance and the total waiting time were weakly correlated in clustered instances with tight time windows and in random instances with wide time windows. In addition, the objectives are in harmony (i.e., the minimization of one tends to also decrease the other) on clustered instances with wide time windows, whereas they are conflicting (i.e., the minimization of one tends to increase the other) on random instances with tight time windows. The total waiting time due to early arrival has also been studied by Zhou and Wang (2014), where a five-objective VRPTW is solved with an objective-wise local search. Moreover, it seems interesting to study together two continuous objectives (here the total traveled distance and the total waiting time) instead of using a discrete objective, like the number of vehicles. Indeed, with two continuous objectives, the fronts generated by the algorithm contain, generally, much more non-dominated solutions, that are relevant when using knowledge discovery methods.

We formalize the notion of waiting time as follows. The waiting time W_i at a customer i is given as the maximum between 0 and the difference between the opening of the TW a_i and the arrival time T_i at location i, that is $W_i = \max\{0, a_i - T_i\}$. Note that each route $r = (v_0, v_1, \ldots, v_{|r|}, v_{|r|+1})$ is associated with a feasible (i.e., consistent with traveling times and TWs) arrival time vector $T_r = (T_{v_0}, T_{v_1}, \ldots, T_{v_{|r|}}, T_{v_{|r|+1}})$ and the total waiting time $W_r(T_r)$ on route r, with respect to T_r is given by $W_r(T_r) = \sum_{i=1}^{|r|} W_{v_i}$. Thus the

total waiting time of a solution $x = \{r_1, \ldots, r_K\}$ on a graph G, given a time arrival vector for each route in the solution, i.e.,$T_x = (T_{r_1}, \ldots, T_{r_K})$, is given by the following formula:

$$f_2(x, T_x) = \sum_{k=1}^{K} W_{r_k}(T_{r_k}) \tag{2}$$

In the remainder of the paper, the problem considered is a Bi-Objective VRPTW (bVRPTW), where functions f_1 (Equation 1) and f_2 (Equation 2) are simultaneously minimized, hence forming a *multi-objective* problem. Key concepts to understand how to solve a multi-objective problem are given in Section 3.

3. MOEA/D Framework

3.1. Multi-Objective Optimization

A *Multi-objective Combinatorial Optimization Problem* (MoCOP) is formalized as follows (Coello et al., 2010):

$$(MoCOP) = \begin{cases} Optimize & F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ s.t. & x \in \mathcal{D}, \end{cases} \tag{3}$$

where n is the number of objectives ($n \geq 2$), x is the vector of decision variables, $\mathcal{D}$ is the (discrete) set of feasible solutions and each objective function $f_i(x)$ has to be optimized (i.e.,minimized or maximized). In multi-objective optimization, the objective function F defines a so-called objective space denoted by $\mathcal{Z}$. For each solution $x \in \mathcal{D}$ there exists a point in $\mathcal{Z}$ defined by $F(x)$.

A *dominance* criterion is defined to compare solutions together: a solution x dominates a solution y, in a minimization context, if and only if for all $i \in [1 \ldots n]$, $f_i(x) \leq f_i(y)$ and there exists $j \in [1 \ldots n]$ such that $f_j(x) < f_j(y)$. The dominance relation introduces a partial order on the solution: there are indeed solutions that cannot be compared among each other.

Then a set of non-dominated solutions is called a *Pareto front*. A feasible solution $x^* \in \mathcal{D}$ is called *Pareto optimal* if and only if there is no solution $x \in \mathcal{D}$ such that x dominates x^*. Resolving a MoCOP involves finding all the Pareto optimal solutions which form the *Pareto optimal set*. The *true Pareto front* of the problem is the image of the Pareto optimal set by the objective function.

Over the years, many metaheuristics based on local search techniques or using evolutionary algorithms (Blot et al., 2018) have been designed to solve multi-objective problems. Moreover, many tools (Riquelme et al., 2015) have been developed to assess and compare the performance of multi-objective algorithms. When the results are compared to a Pareto front of reference, it is common to use the generational distance (GD) metric or the epsilon metric. The GD measures the cumulative (Euclidean) distance from each solution of the front generated to the closest solution in the reference front. The epsilon metric finds the smallest value ϵ, so that, if we multiply the objective vector of all the solutions obtained by ϵ, then the new front strictly dominates the front of reference. However, we do not have access to reference fronts for the bVRPTW studied, which makes it difficult to use these metrics. That is why, in this paper, we use the unary hypervolume (uHV) (Zitzler et al., 2003). It is a metric defined relatively to a reference point Z_{ref}. This indicator evaluates the accuracy, diversity, and cardinality of the front, and it is the only indicator with this capability. Moreover, it can be used without knowing the true Pareto front of the problem. It reflects the volume covered by the members of a non-dominated set of solutions. Thus, the larger the hypervolume, the better the set of solutions.

3.2. Reference MOEA/D

MOEA/D (Zhang and Li, 2007) is a genetic algorithm that approximates the true Pareto front by decomposing the multi-objective problem into M scalar objective subproblems. The scalarization is obtained by weighting each of the n objectives f_k with a weight $w_k \in [0, 1]$, such that $\sum_{k=1}^{n} w_k = 1$. Thus the fitness of a solution x for the subproblem i, defined with a weight $w^i = (w_1^i, w_2^i, \ldots, w_n^i)$, is:

$$f(x|w^i) = \sum_{k=1}^{n} w_k^i \cdot f_k(x) \tag{4}$$

During an iteration, MOEA/D generates a new solution for the i-th subproblem by breeding solutions obtained from *close* (notion defined hereafter) subproblems. The *neighborhood*, of size m, of a weight vector w^i is defined as the set of its m closest (by means of the Euclidean distance) weight vectors among $\{w^1, \ldots, w^M\}$. Then the neighborhood $\mathcal{N}_m(i)$ of the i-th subproblem consists of the m subproblems defined with a weight vector belonging to the neighborhood of w^i. Note that each subproblem is associated with its best solution found during the execution.

At the start of MOEA/D, M uniformly distributed weight vectors are determined and then it proceeds as described in Algorithm 1. For each subproblem, a random initial solution is generated and evaluated, and all these solutions form together the population used in the algorithm. Note that, the size of the population does not change during the execution of the algorithm, and each solution of the population is associated with exactly one subproblem. Each subproblem's neighborhood (of size m) is also computed. When optimizing subproblem i, a random pair of solutions is selected from its neighborhood. The Partially Mapped crossover (PMX), developed by (Goldberg and Lingle, 2014), is applied with probability p_{pmx}, and only one solution is randomly kept. Then, a Local Search (LS), described in Section 4, is applied with probability p_{ls}. Indeed, the mutation is frequently replaced by an LS (Knowles, 2002) in genetic algorithms. Finally, the resulting solution is added to the set S of solutions generated during the iteration, and a few neighbors (at most two in our case) of the subproblem i are updated. When all subproblems have been considered, S is merged with the archive A, containing the non-dominated solutions generated during the execution of the algorithm. It is important to use an external archive since the population may not be able to track all the non-dominated solutions generated due to the discretization of the objective space. If memory issues are encountered, it is common to bound the size of A, but here, with the bVRPTW, it was not necessary. If the termination criterion is reached, the solutions of A are returned, otherwise, a new iteration is started. In the remaining of the paper, this algorithm is our reference algorithm, called $R_{MOEA/D}$.

3.3. Hybrid MOEA/D

3.3.1. Description of the Hybrid Framework

Hybridizing machine learning methods and metaheuristics has become quite common to solve combinatorial problems. In Legrand et al. (2023), MOEA/D is hybridized with a knowledge discovery framework, where the notion of *knowledge groups* is described. The fitness space is divided into k_G regions each representing a *knowledge group*. Each group is characterized by a vector, the reader is referred to the original paper for further details. The region of each group is defined using the subproblems of MOEA/D, in particular, if a solution belongs to the region of a knowledge group, then the knowledge extracted from the solution is added to that group. For this paper, we consider that each subproblem is associated with one group, being the closest one, considering the Euclidean distance between the aggregation of the subproblem and the

Algorithm 1: Reference MOEA/D Framework ($R_{MOEA/D}$).

Input: M weight vectors $w^1, \ldots, w^M$.
Output: The external archive A

`/* Initialisation */`
$(A, S) \leftarrow (\emptyset, \emptyset)$
$P \leftarrow$ random initial population (x^i for the i-th subproblem)
for $i \in \{1, \ldots, M\}$ **do**
    $\mathcal{N}(i) \leftarrow$ indexes of the m closest weight vectors to w^i
    $Obj^i \leftarrow \{f_k(x^i) \,|\, 1 \leq k \leq n\}$
`/* Core of the algorithm */`
while *stopping criterion not satisfied* **do**
    for $i \in \{1, \ldots, M\}$ **do**
        $(i_1, i_2) \leftarrow$ Select $(\mathcal{N}(i))$
        $x \leftarrow \text{PMX}(x^{i_1}, x^{i_2})$
        $x \leftarrow \text{LS}(x)$
        $S \leftarrow S \cup \{x\}$
        updateNeighbors$(P, \mathcal{N}(i), x)$
    $A \leftarrow$ updateArchive(A, S)
    $S \leftarrow \emptyset$
return A

vector of each group. Moreover, the framework relies on two mechanisms, being the knowledge extraction (K_{ext}) and the knowledge injection (K_{inj}). Here, both mechanisms follow an intensification strategy, meaning that for the extraction (resp. injection) the knowledge extracted (resp. injected) belongs to one group only. In the context of the bVRPTW, K_{ext} and K_{inj} are described in Section 3.3.2.

The framework of the hybridization is shown in Algorithm 2. Note that, if lines 3, 11, and 16 are removed, then the algorithm becomes the original MOEA/D described in Section 3.2. At line 3, the procedure `createGroups` creates all the knowledge groups. At line 11, the injection procedure K_{inj} is applied to the current solution x, with probability p_{inj}, using the knowledge stored in the closest group (intensification strategy). At line 16, the extraction procedure K_{ext} is used to extract the knowledge from the set of solutions generated during the iteration. Then it updates the closest group (intensification strategy) of the subproblem being optimized.

3.3.2. Knowledge Mechanisms for the bVRPTW

In this section, the K_{inj} and K_{ext} learning mechanisms are defined to suit the bVRPTW. The mechanisms are inspired by Arnold et al. (2021) who developed PILS, an optimization method relying on frequent patterns from high-quality solutions to explore vast neighborhoods. PILS has already been integrated into the Hybrid Genetic Search (HGS) of Vidal et al. (2014) and the Guided Local Search (GLS) of Arnold and Sörensen (2019) to solve the Capacitated Vehicle Routing Problem (CVRP) with one single objective and provided good results. Usually, patterns are defined as sequences of consecutive customers. Those with a size between 2 and $size_p$, a user-defined parameter, are extracted from generated solutions by K_{ext}. Since the depot belongs to all routes, it is not considered inside patterns. In particular, a route $r = (0, v_1, \ldots, v_{|r|}, 0)$, contains $max(|r| - k + 1, 0)$ patterns of size k. The extracted patterns are added to the corresponding group, where their frequency is incremented. K_{inj} injects a set of N_{Inj} patterns in the solution x provided. Note

Algorithm 2: Hybrid MOEA/D Framework ($H_{MOEA/D}$).

Input: M weight vectors $w^1, \ldots, w^M$. The number $k_{\mathcal{G}}$ of knowledge groups.
Output: The external archive A

`/* Initialisation */`
$(A, S) \leftarrow (\emptyset, \emptyset)$
$P \leftarrow$ random initial population (x^i for the i-th subproblem)
$\mathcal{G} \leftarrow \text{createGroups}(k_{\mathcal{G}})$
for $i \in \{1, \ldots, M\}$ **do**
  $\mathcal{N}(i) \leftarrow$ indexes of the m closest weight vectors to w^i
  $Obj^i \leftarrow \{f_j(x^i) \,|\, 1 \leq j \leq n\}$
`/* Core of the algorithm */`
while *stopping criterion not satisfied* **do**
  for $i \in \{1, \ldots, M\}$ **do**
    $(i_1, i_2) \leftarrow \text{Select}\,(\mathcal{N}(i))$
    $x \leftarrow \text{PMX}(x^{i_1}, x^{i_2})$
    $x \leftarrow \text{K}_{\text{inj}}(x, \mathcal{G}, i)$
    $x \leftarrow \text{LS}(x)$
    $S \leftarrow S \cup \{x\}$
    updateNeighbors$(P, \mathcal{N}(i), x)$
  $A \leftarrow$ updateArchive(A, S)
  $\mathcal{G} \leftarrow \text{K}_{\text{ext}}(\mathcal{G}, S)$
  $S \leftarrow \emptyset$
return A

that, a pattern is kept in the solution only when the solution is improved. To select a pattern, first, its size is randomly chosen among $\{2, \ldots, size_p\}$. Hence the selection is not biased towards smaller, more numerous, patterns. Then the pattern is randomly chosen among the $N_{Frequent}$ most frequent patterns of the same size, which is also a parameter of the algorithm. When all the N_{Inj} patterns have been selected, they are injected one by one according to the following steps. To inject a pattern, arcs connecting it are removed and the nodes are reconnected to form a feasible solution. The reconnection is done optimally, but reversed patterns are not considered due to time windows.

4. Neighborhood Exploration Mechanisms for Local Search

A solution to the problem is represented as a permutation of the customers, and it is evaluated with the split algorithm provided by Prins (2004). For this study, we consider the same neighborhood as defined by Schneider et al. (2017). The applied operators are: Relocate, Swap, and 2-opt*. These operators are commonly used in local search algorithms for routing problems since they are basic operators and they are able to produce a large neighborhood. The relocate operator moves one customer to another location (possibly on the same route). The swap operator exchanges two customers. The 2-opt* operator generalizes the 2-opt (that is an exchange of two arcs in the same route), by involving different routes. The construction of the neighborhood is similar to the one used in the Randomized Variable Neighborhood Descent (see, e.g., Subramanian et al. (2013)), where the order of the neighborhood operators is kept during descent but

shuffled each time the LS is re-applied.

The exploration strategy of the neighborhood (of an operator) is discussed in Section 4.1. Concerning the granular search, used to reduce the neighborhood, different metrics are proposed in Section 4.2 to evaluate how *close* customers are.

4.1. Exploration of the Neighborhood

In this section, more details are given on how the neighborhood of a solution is explored. In local search, the two classical neighborhood exploration strategies are the *best* and the *first* improvement methods (Hoos and Stützle, 2004). Both of them start with the current solution and try to find an improving solution in their neighborhood that is defined by an operator. While the *best* improvement generates all the neighboring solutions of the current solution to find the neighbor with the best improvement, the *first* improvement evaluates the neighboring solutions one by one and stops as soon as it finds one improving neighbor. Obviously, the *best* strategy guarantees the best possible progress at each iteration of the local search but the time allotted to this increases with the size of the problem. On the other hand, the *first* strategy allows a fast exploration but many improving neighbors are set aside. In routing problems, the most commonly used neighborhood exploration strategy is the classical *best* strategy (Subramanian et al., 2013; Schneider et al., 2017; Accorsi and Vigo, 2021), where the best neighboring solution found by the application of one operator on the current solution is selected. The operators presented above (Relocate, Swap, and 2-opt*) provide a number of possible neighbors that depends on the number of customers. Therefore, the neighborhood of a solution increases with the size of the problem. When considering large problems, it may be necessary to speed up each neighborhood exploration. However, the *first* strategy does not consider the best move for a considered customer. A compromise between these two classical strategies called *first-best* has been proposed in the literature for scheduling problems, particularly for permutation flow-shop (Ruiz and Stützle, 2007). Algorithm 3 gives the pseudo-code of the *first-best* procedure. The procedure requires a neighborhood operator (e.g., Swap, Relocate, or 2-opt*), and the solution x which undergoes the LS. For the given operator each customer (l.5) is tentatively moved to its best location until an improving move is found (l.8). When it is the case, the customer is moved, and a new iteration starts. The authorized moves are generated by the procedure `generate_moves` (l.6). The algorithm stops when the whole neighborhood is explored without finding any improving move.

4.2. Neighborhood Metric

Another way to accelerate neighborhood exploration is the reduction of the neighborhood size. In routing problems, it is known that most of the arcs in good solutions involve close customers. Therefore, it may be interesting to consider the *closeness* between customers when we generate the neighboring solutions of a solution. The problem is to define this *closeness*. Indeed, for the classical VRPTW, when minimizing the total traveled distance, the Euclidean distance between customers seems to be enough to quantify the *closeness* between customers. However, when two (or more) objectives are considered, it is not as easy to define a relevant metric. Once a metric is defined, a natural way to reduce the neighborhood of an operator is to consider moves including the δ nearest customers for the metric defined. Roughly speaking, this is the concept behind the *granular search* that was introduced by Toth and Vigo (2003). Having a method that restricts the neighborhood to moves that seem relevant, is interesting to spare time and resources during the LS.

For our study, we compare three different metrics. The first metric, called d_1, is the classical metric

Algorithm 3: The $first\text{-}best$ procedure.

Input: A solution x and a neighborhood operator $\mathcal{N}$
Output: A local optimum

```
improve ← True
while improve do
    improve ← False
    customers ← shuffle([1 ... N])
    for customer ∈ customers do
        moves ← generate_moves(customer, N)
        x' ← best solution obtained by applying a move from moves
        if f(x') < f(x) then
            x ← x'
            improve ← True
            break
return x
```

used in single-objective routing problems: the *closeness* between two customers is simply evaluated as the Euclidean distance between them. The second metric, d_2, is an aggregation between two costs, each one related to one of the objectives. The first cost represents the Euclidean distance between the customers (d_1 in fact), and it is related to the total traveled distance objective. The second cost evaluates the waiting time incurred by going from one customer to another one (not symmetric in general due to the TW). This cost is related to the total waiting time of drivers. More precisely, each subproblem generated in MOEA/D, with weight vector $w = (w_1, w_2)$, has its own metric that is defined as:

$$d_2^w(u,v) = w_1 \cdot \frac{distance(u,v) - MinDistance}{MaxDistance - MinDistance} + w_2 \cdot \frac{WT(u,v) - MinWT}{MaxWT - MinWT} \tag{5}$$

where u and v are two customers, $MaxDistance$ (resp. $MinDistance$) is the largest (resp. smallest) distance between two customers, and $MaxWT$ (resp. $MinWT$) is the largest (resp. smallest) value of WT between two customers (as defined by Equation 6). These values are used to normalize the distances and waiting times, and they are computed at the beginning of the execution since they only depend on the characteristics of the instance. The value $WT(u,v)$ is the waiting time incurred by going to v from u. If $[a_u,\ b_u]$ (resp. $[a_v,\ b_v]$) is the time window of customer u (resp. v), s_u the service time of customer u and t_{uv} the traveling time from u to v, then WT is expressed as follows:

$$WT(u,v) = max(0,\ a_v - (a_u + s_u + t_{uv})) \tag{6}$$

The third and final metric, d_3 is a particular case of d_2 where the weights of the two costs are the same (i.e., 0.5 for both). This metric is interesting because it is independent of the aggregations defined in MOEA/D, and it can therefore be used in another framework, without defining any aggregations.

The two strategies presented in Section 4.1, and the three metrics defined in this section, lead to six variants, being fbd_1, fbd_2, fbd_3, bd_1, bd_2 and bd_3, where fb (resp. b) stands for *first-best* (resp. *best*) strategy and d_X is the metric. These six variants are generalized through the $H_{MOEA/D}$ framework (see Algorithm 2), where the exploration strategy and the metric are parameters of the LS. These variants are

also compared with the reference algorithm, $R_{MOEA/D}$, described in Section 3.2.

5. Experimental Setup

5.1. Benchmarks

Solomon's benchmark (Solomon, 1987) is a set of VRPTW instances, frequently used in the literature to evaluate the performance of multi-objective algorithms (Ghoseiri and Ghannadpour, 2010; Qi et al., 2015; Moradi, 2020). Thus, we use this benchmark to evaluate the performance of all seven variants. The benchmark contains 56 instances, each with 100 customers, divided into three categories according to the type of generation used: R (random), C (clustered), or RC (random-clustered). The R category (23 instances) contains instances where customers are randomly located in a 100×100 grid, while the instances of category C (17 instances) contain clusters of customers. The category RC (16 instances) contains instances where half the customers are randomly located and the other half is clustered. Each category is divided into two classes, either 1 or 2, according to the width of time windows. Instances of class 1 have tighter time windows than instances of class 2, meaning that instances 1 are more constrained. For our experiments, the RC instances are discarded, to highlight the differences in the tuning between random instances and clustered instances.

We also consider bigger instances from the benchmark of Gehring and Homberger (Gehring and Homberger, 1999). This set extends the Solomon one and considers a larger number of customers. Indeed, it contains instances of size 200, 400, 600, 800, and 1000. These instances are very similar to the instances of Solomon since there are the three categories R, C, and RC, and for each category, two classes (1 or 2), depending on the width of the time windows. Contrarily to Solomon's set, each category of instance is equally represented (each one contains 10 instances with tight time windows (class 1) and 10 instances with wide time windows (class 2)) For our experiments, we considered the instances of size 200, without the RC category.

5.2. Termination Criterion and Performance Assessment

Given the size N of an instance (i.e., the number of customers), we set the termination criterion of all the variants to $9 * N^2/125$ seconds. It means that the algorithms are executed during 720 seconds (i.e., 12 minutes) on instances of size 100 and 2880 seconds (i.e., 48 minutes) on instances of size 200. During our preliminary studies, it was enough to obtain accurate and robust results.

The performance of the solution sets is evaluated using the unary hypervolume (Zitzler et al., 2003) (uHV), which quantifies the volume of the region dominated by the solutions in the set. Since the true Pareto front of the problem is unknown, metrics that depend on it (such as epsilon or GD) cannot be utilized. The normalization points for the objectives of each solution are obtained through experiments and are updated dynamically when a new point is discovered. Known optimal values concerning the total traveled distance are available at CVRPLIB. To compute the uHV we normalize the values of the objectives with the normalization points obtained and then we use the point $(1.001, 1.001)$ as a reference. The experiments are run on two computers "Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00GHz", with 24 cores each. The variants have been implemented using the jMetalPy framework (Benitez-Hidalgo et al., 2019). The code is available at https://github.com/Clegrandlixon/data_itor2023.

6. Experimental Protocol

6.1. Tuning

The reference MOEA/D ($R_{MOEA/D}$) uses the following parameters: M, the number of subproblems considered, and m the size of the neighborhood of each subproblem. The probabilities associated with each mechanism are p_{pmx} for the crossover and p_{ls} for the LS. The granularity parameter δ is used to reduce the neighborhood during LS (cf. Section 4.2). The exploration strategy $\mathcal{S}$ (either *best* or *first-best*) and the metric d (either d_1, d_2, or d_3) are also parameters of MOEA/D, and the choice is left to the tuning. The hybrid MOEA/D ($H_{MOEA/D}$) has, in addition, the following parameters: the number of knowledge groups created $k_{\mathcal{G}}$, the probability of injection p_{inj}, the maximal size $size_p$ of the patterns extracted, and the number N_{Inj} of patterns injected, chosen among the $N_{Frequent}$ most frequent patterns. According to a preliminary study (Legrand et al., 2023) and existing works (Arnold et al., 2021), we set $m = 1/4 \times M$ and $N_{Frequent} = 5 \times N_{Inj}$. We propose a different range of values for the eight remaining parameters (cf. Table 1) to define the configuration space in irace.

The tuning is decomposed into three main steps. A first tuning is performed on all the variants, with the whole configuration space defined above. A budget of 2000 is granted to irace, spread between 8 iterations. We generated a set of 20 instances (of size 100) per category to perform the tuning, following the guidelines provided by Uchoa et al. (2017). The generated instances mimic Solomon's instances. Moreover, we separate the tuning of the algorithms on instances R and C, since they do not have the same structure. The tuning of the original MOEA/D allows seeing which neighborhood exploration strategy and metric are better when not performing learning steps. In addition, the tuning of the hybrid framework grants a set of good configurations that can be used to analyze the influence of parameters. Moreover, we perform an additional tuning, where the exploration strategy and the metric are not set in case of learning (i.e., tuning of $H_{MOEA/D}$ directly), allowing us to compare which strategy and metric are better when the learning mechanism is used. See Section 7.1 for more details.

Secondly, we analyze the elite configurations returned for the hybrid variants. Each configuration returned is a local optimum for irace and thus should be the best configuration (statistically) in its close neighborhood. For each configuration, we generate all the possible configurations distant by 1 (with the hamming distance). More precisely, only one parameter is changed at a time (the possible values are the values available in Table 1), while keeping the values of the other parameters unchanged. In order to check whether the configurations returned by irace are good or not, we evaluate the generated configurations on the tuning instances (like before), with the same seed to be fair with the methodology of irace. The hypervolumes obtained are compared in Section 7.2.

In the third part, we use the results of the experiment above to adapt and reduce the configuration space. With the reduction of the space, irace should return more accurate results. Moreover, the number of aggregations (i.e., the number of subproblems) generated is set to 40. By setting this parameter, it is easier to see the impact of the number of groups on the performances of the algorithms. We perform the new tuning of the six variants fbd_1, fbd_2, fbd_3, bd_1, bd_2, and bd_3 on the new space. The budget allocated to irace is reduced accordingly to 1000. The instances are those used for the first tuning. The configurations returned are discussed in Section 7.3.

6.2. Evaluation of Performances

Through our experiments, we investigate the influence of both the strategy of exploration and the metric used during the LS. The algorithms are evaluated with the best configuration returned by irace at the end of

Table 1: Parameter's space given to irace. The granularity and the number of patterns injected are expressed as a percentage of the size of the instance. The parameter representing the number of groups is defined as a percentage of the number of subproblems (when this parameter is equal to 1, only 1 group is created, since 0 groups is not a valid value). The space contains $388\,800$ configurations.

Parameter	Range
Aggregations: M	(20, 40, 60, 80, 100)
Granularity: δ	(10%, 25%, 50%, 75%, 100%)
Probability (PMX): p_{pmx}	(0.00, 0.10, 0.25, 0.50, 0.75, 1.00)
Probability (LS): p_{ls}	(0.00, 0.10, 0.25, 0.50, 0.75, 1.00)
Strategy: $\mathcal{S}$	$\{$*best*, *first-best*$\}$
Metric: d	$\{d_1, d_2, d_3\}$
Number of groups: $k_{\mathcal{G}}$	(1, 10%, 25%, 50%, 75%, 100%)
Maximum size of pattern: $size_p$	(2, 5, 8)
Number of injections: N_{Inj}	(25%, 50%, 75%, 100%)
Probability (Injection): p_{inj}	(0.00, 0.10, 0.25, 0.50, 0.75, 1.00)

the tuning.

In the first experiment, we use Solomon's instances of size 100 (categories R and C), which represent a total of 40 instances. Each variant is executed 30 times on the test instances. For each algorithm, the k-th run of an instance is executed with the same seed, being $10(k-1)$, allowing a fair comparison. We recall that the termination criterion is set to 720 seconds (i.e., 12 minutes) for all variants. For each category of instance (either R or C), we compute the average uHV obtained over the 30 runs. We perform a Friedman test on the average uHV, to know if all algorithms are equivalent, and if it is not the case, we apply a pairwise Wilcoxon test with the *Bonferroni* correction to know which algorithms are statistically better.

A second experiment is performed on Gehring and Homberger's instances of size 200 (on categories R and C too). We keep the same configurations as before (parameters are scaled accordingly, in particular, the granularity and the number of patterns injected). Again, we perform 30 runs on each instance, and each algorithm is executed with the same seeds. The termination criterion is now set to 2880 seconds (i.e., 48 minutes). We perform the same statistical tests to compare the average uHV obtained.

7. Tuning and Parameters

7.1. Preliminary Study

For this set of experiments, we recall, that we performed a tuning of all considered algorithms with irace. The configuration space is given in Table 1. The range of each parameter is chosen to explore various parts of the space. For more details about the tuning, see Section 6.1. The tuning performed by irace on C (resp. R) instances led to the elite configurations stored in Table 2 (resp. Table 3). Only the best elite configuration for each variant is reported (i.e., the configuration of rank 1). The reference algorithm, $R_{MOEA/D}$, seems more efficient with the *first-best* exploration strategy on both categories of instances. The same conclusion is reached for the hybrid one ($H_{MOEA/D}$). Moreover, the metric d_3 seems more appropriate to C instances, and d_2 to R instances, according to the results obtained for $R_{MOEA/D}$ and $H_{MOEA/D}$. In fact, C instances are, in general, more simple to solve. Indeed, the closeness between customers is structurally integrated into the instance, thus a simplified metric, that is d_3, where the waiting time between customers discriminates

easily close customers, seems enough to obtain good results. The metric d_2 is more relevant for R instances, less structured, since the weights between the distance and the waiting time vary according to the aggregation of the subproblem solved. Hence, d_2 discriminates customers according to each aggregated objective. Note that, in neither case the metric d_1 is chosen, meaning that d_1 does not seem adapted to solve these instances in a bi-objective context, as expected.

Concerning the other parameters, we remark that the LS is, in most cases, applied with probability 0.10, which is relatively low, but it is a costly step in terms of execution time. In addition, the granularity remains high (with a value of the 50% or 75% closest customers considered during the search), and the choice of the metric does not seem to have a great impact on it. Moreover, the LS is much more impacting at the beginning of the algorithm, when solutions are bad, thus it may not be interesting to use it too frequently. On the other hand, the injection is applied with a much higher probability (either 0.75 or 1.00). Indeed, the injection operator needs fewer resources and is useful (in terms of diversification and intensification) throughout the execution of the algorithms. During the injection, at least 50 patterns are tentatively injected, and the maximal size of the extracted patterns is almost always set to 5 (surprisingly, it was set to 8 only for the fbd_2 variant on C instances). The value of 5 was the value chosen by the authors of PILS (Arnold et al., 2021). The crossover operator is applied with a probability below 0.50 when using the KD mechanism, i.e., in the hybrid variants ($H_{MOEA/D}$, fbd_1, fbd_2, fbd_3, bd_1, bd_2, and bd_3), whereas it is applied in every iteration with the reference MOEA/D. Indeed, in $R_{MOEA/D}$, the crossover is the only operator that introduces diversity in the solutions, necessary to escape local optima. It is not the case with the other variants, due to the presence of the injection operator. The number of aggregations used is coherent with the values used in general (around 40 aggregations). Only $R_{MOEA/D}$ requires 80 aggregations on R instances to generate better and diverse solutions. Finally, the number of knowledge groups used is very variable, but low values are not used. In particular, strictly more than one groups are created. To be more precise, at least 4 groups are used for each variant. Our main hypothesis concerning the use of many groups is that the structure between two solutions (in terms of patterns extracted) in the objective space may vary significantly, even between solutions with *close* objective values. As a consequence, many groups are created to somehow reduce the *structural gap* between solutions in the same group.

More generally, we observe some differences between the tuning performed for C instances and the tuning performed for R instances. For example, the probability of injection tends to be higher for C instances, while the number of injected patterns tends to be higher for R instances. It validates our choice to tune separately the algorithms on instances R and C. Moreover, many configurations are very similar. Indeed most of them only differ by two or three parameters or use close parameters. In the following, we investigate the influence of the parameters on a subset of elite configurations, in order to see, if the configurations returned are truly local optima in the configuration space provided, and if the configurations can be improved by reducing the space.

7.2. Influence of Parameters on Elite Configurations

Given an elite configuration from Table 2 (resp. Table 3), the set of its neighbors obtained by modifying one parameter (e.g. the number of aggregations) is evaluated on the 20 generated instances of category C (resp. R). The same seed is used for all the evaluations, which is similar to a run performed by irace. The results are reported in Figure 1a (resp. Figure 1b). The (blue) boxplot on the row *Initial* represents the performances of the elite configuration as returned by irace. In other words, it is the reference boxplot. Each boxplot on the rows below represents the performance of the configurations obtained when the corresponding parameter is modified. We remark that only a few parameters have a noticeable impact on the configuration. The most impacting parameters are the strategy ($\mathcal{S}$) used during the LS, the probability of the PMX, and the

Table 2: Best elite configurations returned by irace when tuning on C instances. The symbol "-" reflects the absence of the parameter during the tuning.

Parameter	$R_{MOEA/D}$	$H_{MOEA/D}$	fbd_1	fbd_2	fbd_3	bd_1	bd_2	bd_3
M	40	40	20	40	40	20	40	40
δ (%)	50	75	50	75	50	75	50	50
p_{pmx}	1.00	0.50	0.50	0.25	0.25	0.50	0.10	0.25
p_{ls}	0.10	0.10	0.10	0.10	0.10	0.10	0.25	0.10
$\mathcal{S}$	first-best	first-best	-	-	-	-	-	-
d	d_3	d_3	-	-	-	-	-	-
$k_{\mathcal{G}}$ (%)	-	50	25	10	25	100	10	50
$size_p$	-	5	5	8	5	5	5	5
N_{Inj} (%)	-	100	50	50	75	50	50	75
p_{inj}	-	0.75	1.00	1.00	1.00	1.00	1.00	1.00

Table 3: Best elite configurations returned by irace when tuning on R instances. The symbol "-" reflects the absence of the parameter during the tuning.

Parameter	$R_{MOEA/D}$	$H_{MOEA/D}$	fbd_1	fbd_2	fbd_3	bd_1	bd_2	bd_3
M	80	40	20	60	40	40	40	20
δ (%)	75	75	50	75	75	50	50	50
p_{pmx}	1.00	0.50	0.50	0.50	0.25	0.50	0.25	0.50
p_{ls}	0.10	0.10	0.25	0.10	0.10	0.10	0.10	0.10
$\mathcal{S}$	first-best	first-best	-	-	-	-	-	-
d	d_2	d_2	-	-	-	-	-	-
$k_{\mathcal{G}}$ (%)	-	100	25	100	75	100	10	100
$size_p$	-	5	5	5	5	5	5	5
N_{Inj} (%)	-	100	75	75	75	75	75	75
p_{inj}	-	0.75	1.00	1.00	0.75	0.75	0.75	0.75

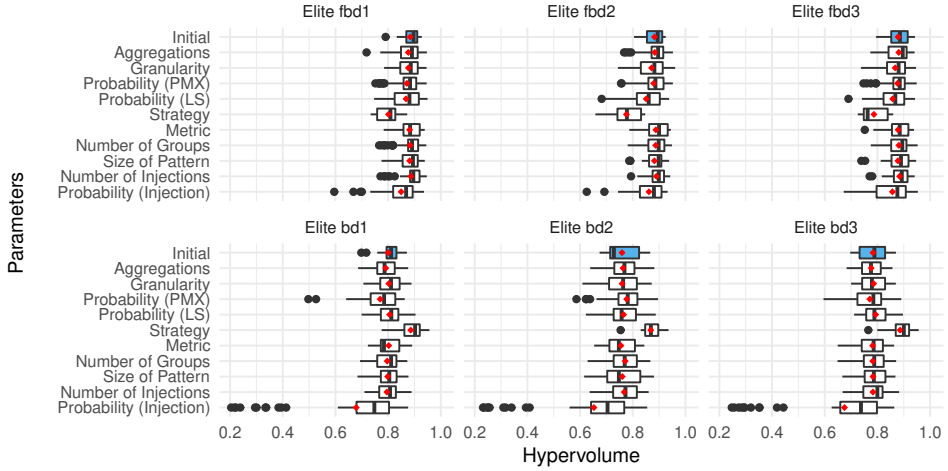
probability of injection.

The strategy only takes two values, and according to the results obtained during the tuning, it seems coherent that changing the *best* strategy to the *first-best* strategy improves the results obtained, while the contrary deteriorates the results. Note that, here, the injection process is activated (all elite configurations have a nonnegative probability of injection).

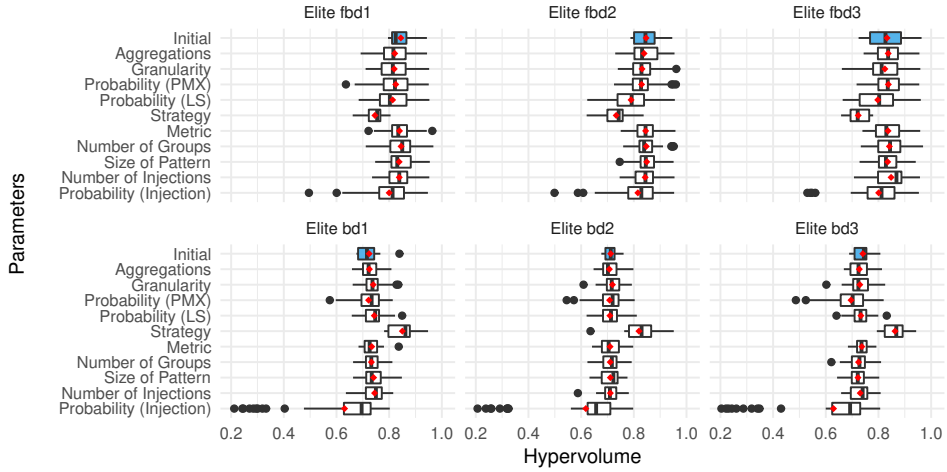
The detailed results for the LS probability are shown in Figure 2. For each possible value of the parameter, the hypervolumes obtained with the corresponding configuration are represented. If nothing really interesting is observed when the strategy is *best*, with the *first-best* one it seems better to consider lower values (between 0.10 and 0.25).

Similarly, the uHV tends to increase when the probability of PMX increases until a plateau is reached after the value of 0.75. The influence of the crossover is even more important with the *best* strategy.

Concerning the probability of injection, the average uHV increases, along with the probability. In particular, when the injection is disabled (probability 0.0), we retrieve the fact that *first-best* strategy performs better than *best* strategy when no learning is used. Moreover, the injection should be applied with high probability (above 0.75) in general. For the other parameters, there are almost no variations of the uHV



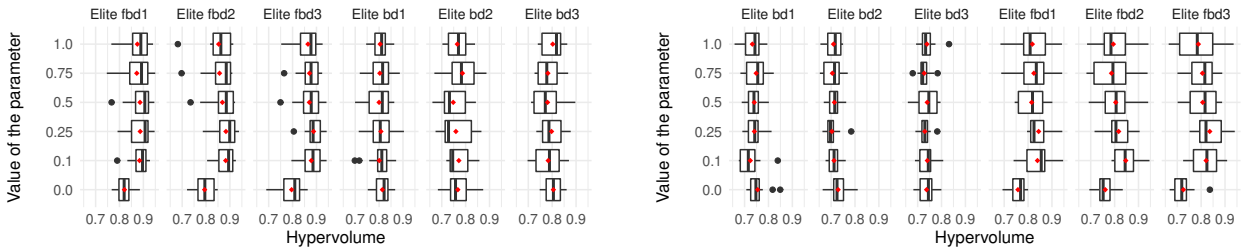
(a) Instances of Category C.



(b) Instances of Category R.

Fig. 1: Performances of elite configurations obtained on (a) C instances and on (b) R instances, when one parameter is modified. The (blue) boxplot refers to the elite configuration as returned by irace.

when the value of the parameter changes. In theory, it means that we could choose any possible value for the remaining parameters, without changing the uHV too much. Moreover, it also explains why the set of elite configurations returned by irace for one variant, contains in general very different values for those parameters. In particular, considering the number of groups, no value seems more interesting than another one to obtain good results. It is very dependent on the other parameters (and in particular, the number of aggregations). We think, that this parameter should also depend on the instance considered since the Pareto



(a) Category C.

(b) Category R.

Fig. 2: Influence of the LS probability on instances of (a) category C and (b) category R. The red dot represents the mean uHV.

Table 4: Reduction of the configuration space. The parameters with a modified range are underlined in red. The space now contains 1 458 different configurations.

Parameter	Range
Aggregations: M	(40)
Granularity: δ	(25%, 50%, 75%)
Probability (PMX): p_{pmx}	(0.25, 0.50, 0.75)
Probability (LS): p_{ls}	(0.10, 0.20, 0.30)
Number of groups: $k_{\mathcal{G}}$	(1, 10%, 25%, 50%, 75%, 100%)
Maximum size of pattern: $size_p$	(2, 5, 8)
Number of injections: N_{Inj}	(100%)
Probability (Injection): p_{inj}	(0.50, 0.75, 1.00)

fronts between two instances can be very different.

7.3. Reduced Configuration Space and Final Tuning

As explained in the former section, some parameters do not have much influence on the hypervolumes returned. That is why we decided to set the number of patterns injected to 100% of the size of the instance (i.e., 100 patterns are tentatively injected on instances of size 100). Note that, it was the value chosen by Arnold et al. (2021) in their own experiments. Moreover, we decided to set the number of aggregations to 40, in order to see if the number of groups used can be more precisely tuned during the tuning. According to what has been said in the previous section, we reduced the range of values for the probability of PMX to $(0.25, 0.50, 0.75)$, and for the probability of injection to $(0.50, 0.75, 1.00)$, being the most interesting values. Concerning the probability of LS, we changed the range to $(0.10, 0.20, 0.30)$, in order to give more adapted parameters to irace. Finally, the granularity is also reduced to $(25, 50, 75)$, since extreme values never belong to elite configurations. All the changes are reported in Table 4.

Using the reduced space of configuration, we performed a new tuning, only on the six variants fbd_1, fbd_2, fbd_3, bd_1, bd_2, bd_3. Indeed, $R_{MOEA/D}$ already had a small configuration space (it has fewer parameters to fix), and the elite configurations returned for $H_{MOEA/D}$ were already in the reduced configuration space defined. The budget allocated to irace is reduced from 2000 to 1000. The best elite configurations are

Table 5: Best elite configurations, from the reduced configuration space, returned by irace on instances of (a) category C, and (b) category R.

(a) Category C.

Param.	fbd_1	fbd_2	fbd_3	bd_1	bd_2	bd_3
δ (%)	50	50	50	50	50	50
p_{pmx}	0.50	0.50	0.50	0.50	0.50	0.50
p_{ls}	0.10	0.10	0.10	0.10	0.10	0.10
$k_{\mathcal{G}}$ (%)	50	50	100	100	100	100
$size_p$	5	5	5	5	2	2
p_{inj}	0.75	1.00	1.00	1.00	0.75	0.75

(b) Category R.

fbd_1	fbd_2	fbd_3	bd_1	bd_2	bd_3
75	75	75	50	50	25
0.50	0.50	0.50	0.50	0.50	0.50
0.10	0.20	0.10	0.10	0.10	0.10
50	25	50	100	100	50
8	8	8	8	8	2
1.00	1.00	0.75	0.75	1.00	0.75

reported in Table 5. At first, we notice that the differences between the two categories of instance are more important, considering the granularity parameter (δ), and the maximum size of patterns extracted ($size_p$). Indeed, the granularity is always set to 50 on C instances, which seems coherent, since these instances contain clusters of customers. The granularity tends to be slightly higher on R instances, except for the variant bd_3, which has the lowest granularity. The maximum size of the patterns is now 8 for most of the variants on R instances, meaning that bigger patterns are interesting when instances are not well structured. For C instances, the maximum size of the pattern remains coherent, since smaller patterns are enough to perform interesting moves.

Another major change concerns the number of knowledge groups used during the execution of the algorithms. The variant being set, more groups are generated on C instances than on R instances. In addition, more groups are generated with strategy *best* than with strategy *first-best*.

Concerning the other parameters, the probability of PMX is set to 0.50 for all variants, the probability of LS is set to 0.10 (except for the variant fbd_2 where it is 0.20), and the probability of injection remains similar to what we saw during the first tuning (see Table2 and Table 3).

8. Experimental Results and Discussion

The first set of experiments is performed on Solomon's instances of size 100 belonging to categories R and C. According to the category, we use the corresponding elite configuration in Table 5. The average uHV obtained through the 30 runs, for each category of instances, is shown in Figure 3 (plot above). The results obtained with the *first-best* strategy are slightly more robust, with smaller boxplots, than those obtained with the *best* one. Moreover, the variant fbd_2 always seems better than the other variants, especially on R instances. Among the six variants, it is the variant bd_3 that returned the worst results.

Table 6 contains the average ranking of the different algorithms on all classes of instances. In this table, the ranks of statistically best algorithms on a class of instances are put in bold. In particular, the variants fbd_2 and fbd_3 are statistically equivalent on $C1$ instances and are better than the other variants. On $C2$ instances, the variants fbd_1, fbd_2, fbd_3, and bd_1 are equivalent. It is not surprising since $C2$ instances are the easiest ones. Moreover, it means that the strategy *best* with the metric d_1, which is usually used in the literature, returns good results when the problem can be modeled by one objective. In addition, all the hybrid MOEA/Ds are better than the reference $R_{MOEA/D}$, which has very low uHV. In fact, $R_{MOEA/D}$ gets easily stuck in local optima, in particular when there must be only one solution on the final front. On R instances, statistically, the best overall algorithm is fbd_2. However, fbd_1 is equivalent to fbd_2 on $R1$ instances. As a

consequence, the strategy *first-best* with the metric d_2, seems to be the best combination, considering all the classes of instances of size 100. This result has already been observed since these parameters were selected for the configuration returned by irace for $H_{MOEA/D}$, i.e. when the exploration strategy and the metric were left as parameters to be tuned (see Table 3). More precisely, the results obtained for all C instances (except C103 and C104) show that the Pareto front contains only one solution, minimizing both objectives. In all the other instances the optimal value of the total traveled distance is not reached by our algorithms, however on some instances (e.g. R101 and R102) it is very close. Concerning the total waiting time, it is always possible to create a solution without waiting time, and our algorithms find them. In addition, we measured the correlation between the objectives on the reference fronts obtained. The reference fronts used, and the associated correlations are available at dataItor2023. Without doing a rigorous analysis of the objective space of the instances like Castro-Gutierrez et al. (2011), we can not say more than some instances should be avoided when studying multi-objective optimization. Table 7 and Table 8 show more detailed results about the convergence of the algorithms fbd_2, bd_2, and $R_{MOEA/D}$. We can remark that, as expected, the final uHV obtained by fbd_2 is higher than the one obtained by bd_2, and $R_{MOEA/D}$. Moreover, fbd_2 is able to reach 80% of the uHV of the reference front significantly faster than $R_{MOEA/D}$ and bd_2 on many instances. When the algorithm does not reach 80% of the uHV at the end of the execution, the maximal time allocated is used instead. It shows that using the learning mechanism with the *first-best* strategy speeds up the convergence process.

The second set of experiments is performed on Gehring and Homberger's instances of size 200. Again, we kept only the instances of categories R and C. The configurations used are the same as the configurations used for the instances of Solomon, except that the granularity is doubled, as well as the number of patterns injected. Indeed, both of these parameters are directly linked to the size of the instance. In Figure 3, we clearly observe that for the benchmark of size 200, the variants using the *first-best* strategy are much more robust than the variants using the *best* one. Moreover, when the exploration strategy is set, there is almost no change when the metric is modified. One hypothesis should be that the given values for the granularity are too high for these instances. Considering the Table 6, the variant fbd_3 is statistically better on instances $C1$ (with tight time windows), while the variants fbd_2 and fbd_3 are statistically equivalent on instances $C2$ (with wide time windows). Moreover, these two algorithms remain better than all the others. On $R1$ instances, we do not reach the same conclusion. Indeed, the variant fbd_1 is statistically better now. However, on $R2$ instances, the three variants fbd_1, fbd_2, and fbd_3 are statistically equivalent and the results returned by these three variants are very close. It is important to notice that with a specific tuning on instances of size 200, we may have reached different conclusions. That being said, strategy *first-best* is always better than strategy *best*. Concerning the metrics, we finally observe that they have a relatively low impact on instances of a bigger size. We meet again the result obtained in Section 7.2, where the metrics did not seem to have an impact on the elite configurations. The reference fronts, the correlations, and additional convergence results are publicly available at dataItor2023.

9. Conclusion

This paper addresses the bi-objective vehicle routing problem with time windows (bVRPTW), a routing problem where both the total traveling cost and the total waiting time of drivers have to be minimized. We chose to solve the bVRPTW with a hybrid MOEA/D (Legrand et al., 2023) where knowledge discovery mechanisms are used to extract patterns, sequences of customers here, and inject them into solutions. Indeed, MOEA/D is known to obtain pretty good performance when solving a bi-objective combinatorial optimization problem. This hybrid MOEA/D framework is modular and both strategical components and parameter values can be fixed. In the mutation step of MOEA/D, local search algorithms are widely used to

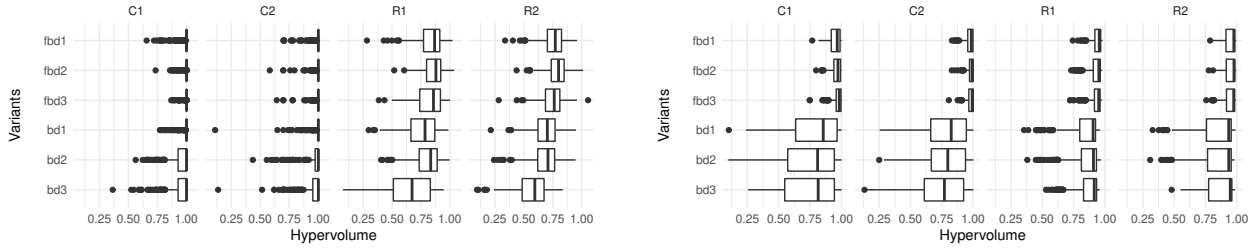


Fig. 3: Results obtained on the different categories of instances, of (left) Solomon's benchmark for size 100, and (right) Gehring and Homberger's benchmark for size 200.

Table 6: Average ranking of all the algorithms on the different categories of instances. Bold ranks are statistically equivalent.

Size	100				200			
Category	C		R		C		R	
Class	C1	C2	R1	R2	C1	C2	R1	R2
$R_{MOEA/D}$	7.0	7.0	6.8	5.3	7.0	4.8	6.8	4.0
fbd_1	3.7	**2.4**	**2.3**	2.3	3.0	2.9	**1.1**	**1.7**
fbd_2	**2.3**	**2.3**	**1.4**	**1.0**	1.9	**1.4**	2.6	**1.7**
fbd_3	**2.2**	**2.9**	2.5	2.7	**1.1**	**1.7**	2.4	**2.6**
bd_1	3.6	**3.3**	5.1	4.7	4.0	5.3	6.1	6.5
bd_2	4.9	4.2	3.9	5.0	5.3	5.6	5.0	6.5
bd_3	4.4	5.9	6.1	7.0	5.7	6.5	4.3	5.0

intensify the search in specific regions and find the best local optimum solutions. While routing problems generally used a local search with a *best* improvement strategy (Hoos and Stützle, 2004) and with a distance metric to reduce the neighborhood size (Schneider et al., 2017), we adapted from scheduling the *first-best* improvement strategy and enhanced the distance metric with the waiting time information of customers. Experiments have been conducted on two benchmarks of the literature. We showed that the *first-best* strategy was better adapted than the *best* strategy, in most cases, to solve bVRPTW instances.

These results are supported by a rigorous tuning of the hybrid MOEA/D and a deep analysis of the influence of its parameters. It emerges from this analysis that only a few parameters are really impacting, like the probability of injection, the probability of crossover, and the probability of local search. Moreover, it is unnecessary to consider the interval $[0, 1]$ entirely, but only a subset containing the most interesting values. The other parameters should either evolve during the execution of the algorithm, in order to adapt to the instances solved (like the granularity, the number of groups, or the number of patterns injected), or be directly set (like the number of aggregations and the maximum size of patterns extracted). This study also allowed us to assess the configurations returned by irace, in particular on the choice of the exploration strategy and the distance metric. Nevertheless, the granularity value returned is higher than expected. Since it does not highly impact the performances of the algorithms, we advise using a smaller value (like 0.25 or 0.50 instead of 0.75) to reduce the memory allocated during the execution. Moreover, the analysis of the neighborhood of elite configurations shows that the configuration space i.e., the values of parameters, was

Table 7: Detailed results for algorithms fbd_2, bd_2, and $R_{MOEA/D}$ on C instances. From left to right: the average size of the front, the average uHV, and the average time to reach 80% of the reference uHV.

	fbd$_2$			bd$_2$			$R_{MOEA/D}$		
Inst.	$\|F\|$	uHV	Time (s)	$\|F\|$	uHV	Time (s)	$\|F\|$	uHV	Time (s)
C101	1.0	**1.002**	**57.3**	1.0	**1.002**	197.1	1.8	0.354	682.7
C102	1.1	**0.988**	**120.7**	1.0	**0.980**	362.6	3.2	0.033	720.3
C103	1.5	**0.984**	**211.6**	2.0	0.712	614.6	5.3	0.000	720.4
C104	1.7	**0.905**	**316.4**	2.8	0.588	702.9	4.5	0.000	720.4
C105	1.0	**0.989**	**111.0**	1.0	**0.989**	248.7	1.9	0.068	715.2
C106	1.0	**1.002**	**69.0**	1.0	**1.002**	255.5	1.9	0.208	720.4
C107	1.0	**0.972**	**140.2**	1.0	**0.962**	324.6	1.5	0.013	720.2
C108	1.0	**0.971**	**158.4**	1.0	0.925	424.8	1.4	0.001	720.3
C109	1.0	**0.944**	**191.5**	1.0	0.802	505.5	1.5	0.000	720.3
C201	1.0	**1.002**	**33.8**	1.0	**1.002**	171.0	1.2	0.948	205.0
C202	1.0	0.994	**97.4**	1.0	**1.002**	308.1	1.0	0.743	573.3
C203	1.0	**0.937**	**106.4**	1.1	0.890	404.0	1.1	0.704	529.1
C204	1.0	**0.849**	**333.4**	1.4	0.664	658.9	1.5	0.450	657.7
C205	1.0	**0.990**	**47.2**	1.0	0.973	259.8	1.1	0.865	345.0
C206	1.0	**1.002**	**55.6**	1.0	0.978	372.1	1.0	0.821	481.5
C207	1.0	**1.002**	**49.5**	1.0	0.970	332.0	1.1	0.749	487.9
C208	1.0	**1.002**	**56.3**	1.1	0.987	371.7	1.3	0.751	507.6

well defined since a single parameter cannot significantly change the performance obtained on the tuning instances.

In future works, we will investigate the adaptive side of the algorithm by automatically updating some parameters during the execution of the algorithm. Besides, we will tackle different variants of the bVRPTW to validate or not our conclusions when the objectives are modified.

Acknowledgements

This work has been supported by the French National Research Agency through the AI_PhD@Lille program (ANR-20-THIA-0014). This support is gratefully acknowledged.

References

Accorsi, L., Vigo, D., 2021. A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transportation Science* 55, 4, 832–856.

Arnold, F., Santana, Í., Sörensen, K., Vidal, T., 2021. PILS: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recognition*

Arnold, F., Sörensen, K., 2019. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research* 105, 32–46.

Baños, R., Ortega, J., Gil, C., Márquez, A.L., De Toro, F., 2013. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers & Industrial Engineering* 65, 2, 286–296.

Benitez-Hidalgo, A., Nebro, A.J., Garcia-Nieto, J., Oregi, I., Del Ser, J., 2019. jmetalpy: A python framework for multi-objective

Table 8: Detailed results for algorithms fbd_2, bd_2, and $R_{MOEA/D}$ on R instances. From left to right: the average size of the front, the average uHV, and the average time to reach 80% of the reference uHV.

	fbd$_2$			bd$_2$			$R_{MOEA/D}$		
Inst.	$\|F\|$	uHV	Time (s)	$\|F\|$	uHV	Time (s)	$\|F\|$	uHV	Time (s)
R101	72.2	**0.915**	**65.0**	70.6	**0.911**	229.7	47.7	0.907	73.4
R102	45.6	**0.923**	**137.8**	45.0	**0.911**	296.8	27.1	0.876	232.7
R103	23.7	**0.937**	**277.8**	23.4	0.899	457.0	20.3	0.724	721.1
R104	6.0	**0.915**	**341.2**	6.3	0.881	533.4	6.4	0.499	721.2
R105	12.3	**0.941**	**166.2**	13.9	0.906	299.6	7.7	0.862	438.6
R106	6.2	**0.894**	**258.8**	8.0	**0.878**	406.5	5.5	0.763	700.1
R107	6.0	**0.851**	**496.2**	5.1	0.828	602.9	6.8	0.524	720.9
R108	1.8	**0.880**	**358.2**	1.5	0.851	532.8	2.5	0.389	720.9
R109	1.2	**0.915**	**259.5**	1.4	0.877	450.8	1.5	0.727	680.3
R110	1.1	**0.843**	**368.9**	1.3	**0.849**	496.6	1.3	0.637	717.1
R111	1.9	**0.850**	**423.1**	1.9	0.809	583.4	2.4	0.517	720.7
R112	1.0	**0.845**	**480.2**	1.0	0.804	559.9	1.0	0.387	720.4
R201	52.1	**0.854**	**230.3**	52.2	**0.843**	376.4	33.6	0.814	525.9
R202	44.7	**0.857**	**311.6**	42.1	**0.849**	457.3	25.8	0.824	515.1
R203	34.5	**0.859**	**338.4**	31.2	0.819	600.0	21.3	0.797	613.9
R204	13.2	**0.806**	**548.7**	11.9	0.752	679.7	8.1	0.719	706.6
R205	20.6	**0.839**	**413.7**	22.3	**0.833**	509.6	14.4	0.798	637.2
R206	22.5	**0.867**	**349.4**	22.9	0.839	534.9	13.0	0.821	581.8
R207	19.4	**0.828**	**467.1**	16.4	0.805	622.5	10.7	0.776	680.6
R208	5.5	**0.808**	**541.6**	6.5	0.790	605.5	3.8	0.720	718.8
R209	12.0	**0.808**	**543.2**	12.6	0.776	648.0	10.0	0.755	712.1
R210	20.4	**0.855**	**368.3**	20.5	0.830	555.0	12.3	0.791	649.3
R211	1.0	**0.786**	**548.7**	1.0	**0.768**	618.9	1.0	0.732	673.5

optimization with metaheuristics. *Swarm and Evolutionary Computation* 51, 100598.

Blot, A., Marmion, M., Jourdan, L., 2018. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *J. Heuristics* 24, 6, 853–877.

Castro-Gutierrez, J., Landa-Silva, D., Pérez, J.M., 2011. Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE.

Coello, C.A.C., Dhaenens, C., Jourdan, L., 2010. Multi-objective combinatorial optimization: Problematic and context. In *Advances in multi-objective nature inspired computing*. Springer, pp. 1–21.

Gehring, H., Homberger, J., 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, Vol. 2, Springer Berlin, pp. 57–64.

Geiger, M., 2003. A computational study of genetic crossover operators for multi-objective vehicle routing problem with soft time windows. In *Multi-Criteria-und Fuzzy-Systeme in Theorie und Praxis*. Springer, pp. 191–207.

Ghoseiri, K., Ghannadpour, S.F., 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing* 10, 4, 1096–1107.

Goldberg, D.E., Lingle, R., 2014. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, Psychology Press, pp. 154–159.

Hoos, H.H., Stützle, T., 2004. *Stochastic local search: Foundations and applications*. Elsevier.

Jozefowiez, N., Semet, F., Talbi, E.G., 2008. Multi-objective vehicle routing problems. *European Journal of Operational Research* 189, 2, 293–309.

Knowles, J.D., 2002. Local-search and hybrid evolutionary algorithms for Pareto optimization. Ph.D. thesis, University of Reading,

Reading.

Legrand, C., Cattaruzza, D., Jourdan, L., Kessaci, M.E., 2022. Enhancing MOEA/D with learning: Application to routing problems with time windows. In *Proceedings of the GECCO companion*, pp. 495–498.

Legrand, C., Cattaruzza, D., Jourdan, L., Kessaci, M.E., 2023. Improving moea/d with knowledge discovery. application to a bi-objective routing problem. In *EMO: 12th International Conference, 2023, Proceedings*, Springer, pp. 462–475.

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

Melián-Batista, B., De Santiago, A., Angel-Bello, F., Alvarez, A., 2014. A bi-objective vehicle routing problem with time windows: A real case in tenerife. *Applied Soft Computing* 17, 140–152.

Moradi, B., 2020. The new optimization algorithm for the vehicle routing problem with time windows using multi-objective discrete learnable evolution model. *Soft Computing* 24, 9, 6741–6769.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research* 31, 12, 1985–2002.

Qi, Y., Hou, Z., Li, H., Huang, J., Li, X., 2015. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. *Computers & Operations Research* 62, 61–77.

Riquelme, N., Von Lücken, C., Baran, B., 2015. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, IEEE, pp. 1–11.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European journal of operational research* 177, 3, 2033–2049.

Schneider, M., Schwahn, F., Vigo, D., 2017. Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research* 263, 2, 493–509.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 2, 254–265.

Subramanian, A., Uchoa, E., Ochi, L.S., 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research* 40, 10, 2519–2531.

Toth, P., Vigo, D., 2003. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing* 15, 4, 333–346.

Toth, P., Vigo, D., 2014. *Vehicle routing: problems, methods, and applications*. SIAM.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A., 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research* 257, 3, 845–858.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*

Zhang, K., Cai, Y., Fu, S., Zhang, H., 2019. Multiobjective memetic algorithm based on adaptive local search chains for vehicle routing problem with time windows. *Evolutionary Intelligence* 24, 2, 1–12.

Zhang, Q., Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation* 11, 6.

Zhou, Y., Wang, J., 2014. A local search-based multiobjective optimization algorithm for multiobjective vehicle routing problem with time windows. *IEEE Systems Journal* 9, 3, 1100–1113.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G., 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation* 7, 2, 117–132.