Enhancing MOEA/D with Learning: Application to Routing Problems with Time Windows

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

Enhancing MOEA/D with Learning: Application to Routing Problems with Time Windows

Clément Legrand
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL
F-59000 Lille, France
clement.legrand4.etu@univ-lille.fr

Diego Cattaruzza
Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL
F-59000 Lille, France
diego.cattaruzza@centralelille.fr

Laetitia Jourdan
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL
F-59000 Lille, France
laetitia.jourdan@univ-lille.fr

Marie-Eléonore Kessaci
Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL
F-59000 Lille, France
marie-eleonore.kessaci@univ-lille.fr

ABSTRACT

Integrating machine learning (ML) techniques into metaheuristics is an efficient approach in single-objective optimization. Indeed, high-quality solutions often contain relevant knowledge, that can be used to guide the heuristic towards promising areas. In multi-objective optimization, the quality of solutions is evaluated according to multiple criteria that are generally conflicting. Therefore, the ML techniques designed for single-objective optimization can not be directly adapted for multi-objective optimization. In this paper, we propose to enhance the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) with a clustering-based learning mechanism. To be more precise, solutions are grouped regarding a metric based on their quality on each criterion, and the knowledge from the solutions of the same group is merged. Experiments are conducted on the multi-objective vehicle routing problem with time windows. The results show that MOEA/D with learning outperforms the original version.

CCS CONCEPTS

• **Mathematics of computing** → **Combinatorial optimization**;
• **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

Machine Learning, Hybridization, Multi-objective optimization, Operations research, Routing and layout

ACM Reference Format:
Clément Legrand, Diego Cattaruzza, Laetitia Jourdan, and Marie-Eléonore Kessaci. 2022. Enhancing MOEA/D with Learning: Application to Routing Problems with Time Windows. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528909

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528909

1 INTRODUCTION

Machine Learning (ML) has become incredibly popular in the past few years, especially in the field of optimization [13]. Indeed, it is known that optimization and learning have a good synergy [7]. In most cases, the learning consists of an extraction step, where the mechanism discovers knowledge from previously generated solutions, and an injection step, which exploits the extracted knowledge to guide the algorithm towards promising areas. The knowledge can take different forms, like parts of the structure of the solutions. The design of efficient learning tools is an active research domain [3].

Due to the success of the integration of learning into single-objective problems, it seems natural to consider the integration of ML into multi-objective combinatorial optimization problems (MoCOPs) [6]. Such problems are frequent in the industry where decision-makers are interested in optimizing several conflicting objectives at the same time. The objectives can be of different nature (economical, ecological, ethical), and allow to consider the problem from different points of view. Concerning the design of learning tools for MoCOPs, the literature is parse. In a MoCOP, the quality of solutions is evaluated according to multiple criteria that are generally conflicting. Such an evaluation prevents the definition of a total order on solutions, meaning that some solutions are incomparable. Therefore, most of ML techniques designed for single-objective problems can not directly be used for MoCOPs.

In this article, we propose a hybridization between MOEA/D and a clustering-based learning mechanism to solve a bi-objective Vehicle Routing Problem with Time Windows (VRPTW) [14]. Indeed, when solutions can be represented as permutations, like here, it is possible to learn sequences of elements, called patterns, inside the permutation. MOEA/D [18] is a widely studied algorithm [17]. The learning mechanism extracts the knowledge from close solutions in the objective space, by creating groups of solutions.

To summarize, the contribution of this paper is the design of a learning MOEA/D that is efficient to solve a bi-objective VRPTW.

The remaining of the paper is organised as follows. In Section 2 multi-objective problems are briefly introduced, and MOEA/D is presented. Section 3 focuses on learning and optimization, and our learning variant of MOEA/D. Section 4 describes the problem studied and the knowledge extracted. Section 5 presents the experimental protocol and then gives and discusses the experimental results. Finally, Section 6 concludes and presents perspectives for this work.

2 MULTI-OBJECTIVE OPTIMIZATION

2.1 Multi-Objective Problems

Many real-world problems may be modeled by two or more objectives (often conflicting), that should be simultaneously optimized. Considering logistic problems, for instance, many challenges have to be tackled (economical and environmental). In the following MoCOPs are briefly presented. The reader may refer to [6] for a complete formalization of MoCOPs.

The functions $f_1, \ldots, f_n$ refer to the $n \geq 2$ objectives that have to be optimized. For each feasible solution x, there exists a point in the objective space, noted $\mathcal{Z}$, defined by $F(x) = (f_1(x), \ldots, f_n(x))$.

It is possible to compare some solutions through a *dominance* criterion. A solution x dominates a solution y (noted $x \prec y$), 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 $f_j(x) < f_j(y)$. However, this relation only defines a partial order on the solution set.

Then a set of non dominated solutions is called a *Pareto front*. A feasible solution x^* is called *Pareto optimal* if and only if there does not exists any solution x 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 obtained by plotting the objective function values corresponding to the solutions in the Pareto optimal set.

Over the years, many metaheuristics with local search techniques or using evolutionary algorithms [5] have been designed to solve multi-objective problems.

Moreover, many tools have been developed to assess and compare the performance of multi-objective algorithms. In this paper we use the unary hypervolume (HV) [19], defined relatively to a reference point Z_{ref}. This indicator evaluates accuracy, diversity and cardinality of the front, and it is the only indicator with this capability. It reflects the volume in the objective function space covered by the members of a non dominated set of solutions. Hence, the larger the hypervolume, the better the set of solutions. This metric can be computed without knowing the true Pareto front of the instance, that is interesting for our study.

2.2 MOEA/D

MOEA/D [18], is a genetic algorithm that approximates the Pareto front by decomposing the multi-objective problem into several scalar objective subproblems. Each iteration of the algorithm optimizes one of the subproblems, by applying a genetic step composed of a crossover and a mutation operator.

Here, we consider scalar problems obtained with a weighted sum of the objectives. More precisely a convex combination of the n objectives is defined by attributing a weight $w_i \in [0, 1]$ to the objective f_i such that $\sum_{i=1}^{n} w_i = 1$. Then the fitness of a solution is the following quantity: $g(x|w) = \sum_{i=1}^{n} w_i \cdot f_i(x)$. The solution which minimizes this fitness is Pareto optimal [10], regardless of the structure of the true Pareto front. Thus to generate different Pareto optimal solutions one can use M different weight vectors $w^1, \ldots, w^M$ in the expression of the fitness.

MOEA/D minimizes the i-th subproblem, by using the solutions of its closest neighbors. Indeed, the *neighborhood*, of size m, of a weight vector w^i is defined as the set of its m closest (for the euclidean distance) weight vectors in $\{w^1, \ldots, w^M\}$. Then the neighborhood $\mathcal{N}_m(i)$ of the i-th subproblem simply consists of the m subproblems defined with a weight vector belonging to the neighborhood of w^i. In the following we consider a uniform distribution on the weight vectors, and we assume that is enough to obtain diverse subproblems.

During the execution of MOEA/D, only the best solution found is kept for each subproblem. When a subproblem i is optimized, the genetic step generates a new solution. The crossover occurs with probability p_{cro} and the mutation with probability p_{mut}. Note that, the crossover is realized between two solutions from subproblems of $\mathcal{N}_m(i)$. Moreover an external archive is used to store nondominated solutions found during the search. These solutions are returned once the termination criteria is reached.

Since the algorithm is applied on a problem where solutions are permutations, the crossover applied is the partially mapped crossover (PMX), and the mutation is a permutation swap, that exchanges two elements of a permutation.

3 INTEGRATING LEARNING INTO MOEA/D

3.1 Learning from Solutions in Optimization

Hybridizing machine learning methods and metaheuristics is quite common to solve combinatorial problems. Indeed the survey [13] reviews different kind of hybridizations and proposes to classify the different methods according to where the hybridization is performed: at a problem-level, at a low-level or at a high-level. When learning is integrated at a low-level, the knowledge is extracted from solutions of the problem (e.g. to learn their structure). This is the kind of integration we are interested in. In particular, many ML methods can be used, and here we investigate *clustering*, to group the solutions, and *association rules* (patterns to be more precise), to learn from solutions.

Moreover the integration can be realized either *online* or *offline* [7]. The learning is said online when it uses resources generated during the execution. Otherwise the learning is said offline.

Most of the learning mechanisms are composed of an *extraction* step, where something is learned, and an *injection* step, which uses the extracted knowledge to find new promising solutions. The following section, presents the integration of learning into MOEA/D.

3.2 Learning within MOEA/D and Variants

We base our learning mechanism for MOEA/D on the following fact: close solutions in the objective space may have the same structure. Note that, the *closeness* between two solutions is evaluated according to their objective vector with the euclidean distance. That is why, gathering the knowledge from neighboring solutions in the objective space is relevant. To that aim, we introduce the notion of *learning groups*.

Each learning group $\mathcal{G}$ is associated to one of the subproblems defined in MOEA/D. More precisely $\mathcal{G}_i$ is associated to the subproblem of weight vector w^i. Thus, there are as many learning groups as subproblems. Moreover, since the neighborhood of each subproblem is already defined in MOEA/D, we keep the same neighborhood for the learning groups. In other words, if each subproblem has m neighbors, then each solution will belong to m learning groups.

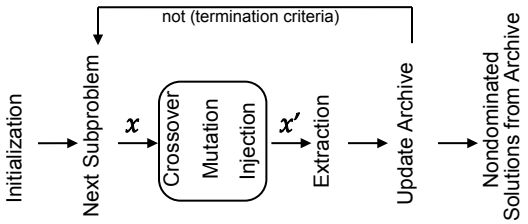

Figure 1: The MOEA/D learning framework.

With this construction, our will is that each learning group focuses on a part of the objective space. The Figure 1 shows the main steps of the algorithm, and the pseudo-code is available as supplementary material. We present now, the main steps of the algorithm.

MOEA/D is initialized, with a uniform set of weight vectors, and an initial population of randomly generated solutions. Initially, each learning group is empty. Then the algorithm iterates over the subproblems defined. Suppose that the i-th subproblem is optimized. The crossover is the first operator applied, with probability p_{cro}. Two neighboring subproblems are selected that give two solutions to perform the PMX crossover. Then, the two solutions undergo a permutation swap mutation, with probability p_{mut}. Only the best solution x_b among the offspring is considered to undergo the injection step, since it is a costly step. The injection is performed with probability p_{inj}. Here we distinguish between two possible strategies leading to two hybridization models. The first one, called H_{int}-MOEA/D, uses an intensification strategy for the injection, to focus on a specific part of the objective space. In this case, x_b receives the knowledge contained in the learning group $\mathcal{G}_i$ that is to say the learning group associated to its own subproblem. The second one, called H_{div}-MOEA/D, uses a diversification strategy, to favour a larger exploration of the space. This time x_b receives the knowledge contained in a learning group, randomly chosen among all the existing groups. Finally, once the injection produced a solution x', the extraction step occurs with a probability p_{ext}. The knowledge of x' is extracted and is added to the learning groups of the neighboring subproblems.

4 APPLICATION

4.1 Multi-Objective VRPTW

The MO-VRPTW [14] considered in this work 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 in a travel cost c_{ij} and a travel time t_{ij}. Vertex 0 represents the depot where a fleet of K identical vehicles with limited capacity Q is based. Vertices $1, \ldots, N$ represent the customers to be served, each one having a demand q_i and a time window $[a_i, b_i]$ during which service must occur. Vehicles may arrive before a_i. In that case the driver has to wait until a_i to accomplish service incurring in a waiting time. Arriving later than b_i is not allowed. It is assumed that all inputs are nonnegative integers. The MO-VRPTW calls for the determination of at most K routes such that the travelling cost and waiting time are simultaneously 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 Q, (d) time windows are respected. The solutions of this problem are represented as permutations of customers, and are evaluated with the split algorithm [11].

A bi-objective VRPTW, where the number of vehicles is minimized instead of the waiting time has been studied by Ghoseiri et al. [8]. To solve the problem they proposed a genetic algorithm.

4.2 Pattern Injection Local Search

In the field of routing problems, a learning mechanism called Pattern Injection Local Search (PILS), has recently been introduced by Arnold et al. [1]. This mechanism is an optimization strategy that uses frequent patterns from high-quality solutions, to explore high-order local-search neighborhoods. PILS has been hybridized with the Hybrid Genetic Search (HGS) of Vidal et al. [16] and the Guided Local Search (GLS) of Arnold and Sörensen [2] to solve the Capacitated Vehicle Routing Problem (CVRP) with good results.

PILS extracts patterns (i.e. sequences of customers) from solutions. The patterns have a size between 2 and $MaxSize$, a user defined parameter. For the injection step the reader is refered to the article of Arnold et al. [1]. This step brings diversity to the solution by injecting some frequent patterns learned. Note that, there is a minor change with the mechanism described in the article: reversed patterns are not considered because of time windows.

5 EXPERIMENTS

5.1 Experimental Protocol

5.1.1 Experimental Setup. We compare three algorithms: MOEA/D, as presented in Section 2.2, and the two hybrid-MOEA/Ds described in Section 3.2: H_{int}-MOEA/D and H_{div}-MOEA/D. The knowledge is extracted and injected as presented in Section 4.2.

We use the Solomon's instances [12] to evaluate the performances of MOEA/D and the two hybrid-MOEA/Ds. These instances were designed for the single-objective VRPTW, but they are also used to evaluate the performances of multi-objective algorithms [8]. The benchmark contains 56 instances divided into three categories according to the type of generation used, either R, C or RC. The generation R randomly places customers in the grid, while the generation C tends to create clusters of customers. Each category is itself divided into two classes $1XX$ or $2XX$ according to the width of time windows. Instances of class $1XX$ have wider time windows than instances of class $2XX$, meaning that instances $2XX$ are more constrained. The generation RC mixes both generations. There exists instances of size 25, 50 and 100, however instances of size 25 and 50 are restrictions of instances of size 100. We do not consider instances of size 25 since they are too small.

In order to be fair, we tune the three algorithms with irace [9] to evaluate and compare the performances of their best version. However, the instances used for tuning have to be different from the ones used to evaluate the algorithms. Thus, we generated 96 new instances of sizes 50 and 100, by using the method described by Uchoa et al. [15], to mimic the Solomon's instances.

We run the experiments on two computers "Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00GHz", with 24 cores each, in parallel (using slurm). The implementation of the hybrid-MOEA/Ds has been realized with the jMetalPy framework [4].

Variant	R1	R2	RC1	RC2	C1	C2
H_{int}	51.8	69.6	47.4	73.5	75.7	114.8
H_{div}	**52.6**	**72.2**	**47.7**	**75.5**	**75.9**	**115.0**

Table 1: Average gain (%) obtained with both hybrids with respect to HVs returned by MOEA/D on instances of size 100.

Variant	R1	R2	RC1	RC2	C1	C2
MOEA/D	31.1	68.5	35.8	79.2	96.1	182.9
H_{int}	2.1	8.2	**2.6**	6.0	0.1	0.4
H_{div}	**1.6**	**4.3**	**2.6**	**3.6**	**0.0**	**0.0**

Table 2: Best gaps (%) obtained for the total cost objective, relatively to the best-known on instances of size 100.

5.1.2 Performance Assessment. The parameters obtained are available as supplementary material. Now, the performances of each algorithm are compared over the Solomon's instances. The three algorithms share the same maximum running time allocated, set to $6 \times N$ seconds, allowing hypervolume-convergence for all. The algorithms are executed 30 times on each instance. The k-th run of an instance is executed with the seed $10 \times (k-1)$, so that, all algorithms are compared with the same seeds.

Once all the tests done, the Pareto fronts are normalized according to the best and worst objectives obtained. They are compared with the hypervolume metric, since we do not know the true Pareto fronts of the instances. The reference point used to compute the hypervolume is $(1.001, 1.001)$. The results are reported in Section 5.2.

5.2 Results and Discussion

Table 1 shows the average gain obtained on instances of size 100, with the hybrid-MOEA/Ds, when MOEA/D is the reference algorithm. Similar results are obtained on instances of size 50. The detailed results are available as supplementary material. We can see that there is always a mean gain of at least 47.4%. Meaning that, H_{int} and H_{div} return much better hypervolumes than MOEA/D. The biggest gains are obtained on clustered instances.

If we compare H_{int} and H_{div} variants, one can see that they have a similar gain on the same instances. The pairwise Wilcoxon tests, showed that both algorithms are equivalent on most instances. But H_{div} returns always slightly better results than H_{int}, illustrating the importance of bringing diversity during the injection step. The Table 2 shows the best gaps obtained between the best-known and our solutions, relatively to the cost objective. The best-known are taken from vrp-rep and, if the optimal is not available, from [8]. More detailed tables are available as supplementary material. This table highlights the performances of the learning mechanism, and more particularly of the H_{div} variant. In particular H_{div}, returns the optimal cost on C instances, and it is able to return new solutions on $RC2$ instances. Note that, few optimal solutions are available for these instances.

6 CONCLUSION AND PERSPECTIVES

In this paper we presented an enhanced version of MOEA/D using a learning mechanism. This learning mechanism creates learning groups to gather the knowledge of quality-close solutions. Through our experiments, we showed that the hybridization works successfully on a bi-objective routing problem with time windows, meaning that it is worth to spend time for learning. Moreover the variant H_{div}-MOEA/D, is able to produce competitive results on some instances, meaning that bringing diversity during the search is a crucial step. However, some results obtained have a high gap ($> 1\%$) with the best-known, meaning that the algorithm can be further improved.

As future works, we would replace the mutation operator with a local search, which should globally help MOEA/D, but it may also improve the quality of the learning, since local optima tend to provide more reliable sequences [1].

REFERENCES

[1] Florian Arnold, Ítalo Santana, Kenneth Sörensen, and Thibaut Vidal. 2021. PILS: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recognition* 116 (2021), 107957.

[2] Florian Arnold and Kenneth Sörensen. 2019. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research* 105 (2019), 32–46.

[3] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2020. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research* (2020).

[4] Antonio Benitez-Hidalgo, Antonio J Nebro, Jose Garcia-Nieto, Izaskun Oregi, and Javier Del Ser. 2019. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation* 51 (2019), 100598.

[5] Aymeric Blot, Marie-Eléonore Marmion, and Laetitia Jourdan. 2018. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *J. Heuristics* 24, 6 (2018), 853–877.

[6] Carlos A Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan. 2010. Multi-objective combinatorial optimization: Problematic and context. In *Advances in multi-objective nature inspired computing*. Springer, 1–21.

[7] David Corne, Clarisse Dhaenens, and Laetitia Jourdan. 2012. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research* 221, 3 (2012), 469–479.

[8] Keivan Ghoseiri and Seyed Farid Ghannadpour. 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing* 10, 4 (2010), 1096–1107.

[9] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[10] Kaisa Miettinen. 2012. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media.

[11] Christian Prins. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research* 31, 12 (2004).

[12] Marius M Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 2 (1987).

[13] El-Ghazali Talbi. 2020. Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics. (2020).

[14] Paolo Toth and Daniele Vigo. 2014. *Vehicle routing: problems, methods, and applications*. SIAM.

[15] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. 2017. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research* 257, 3 (2017), 845–858.

[16] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* (2014).

[17] Qian Xu, Zhanqi Xu, and Tao Ma. 2020. A survey of multiobjective evolutionary algorithms based on decomposition: variants, challenges and future directions. *IEEE Access* 8 (2020), 41588–41614.

[18] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation* 11, 6 (2007), 712–731.

[19] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation* 7, 2 (2003), 117–132.